리눅스 NET-3-HOWTO, Linux Networking.

저자 : Terry Dawson, VK2KTJ, *terry@perf.no.itg.telstra.com.au* `<mailto:terry@perf.no.itg.telstra.com.au>`

일시 : 1997년 7월 12일 (2/3 분량)

역자 : 조용준 , *sudoer@nownuri.net* `<mailto:sudoer@nownuri.net>` *vandal@danjae.skku.ac.kr* `<mailto:vandal@danjae.skku.ac.kr>`

리눅스 오퍼레이팅 시스템은 거의 대부분이 scratch로 작성된 커널기반의 네트 워킹 지원을 자랑스럽게 여긴다. 최근 커널에 포함되어 있는 tcp/ip implementa tion의 성능은 비슷한 수준의 도구에 대해서도 그것을 가치있는 것으로 만들어 준다. 이 문서는 리눅스 네트워킹 소프트웨어, 관련된 툴들의 설치와 설정방법을 기술 하는데 목표를 두고있다.

목 차

제 1 절 이전 버전과 달라진 것. 3

제 2 절 소개 3

2.1 Feedback . . . 3

제 3 절 이 문서를 어떻게 사용할 것인가(net-3-howto-howto?). 3

제 4 절 General Information about Linux Networking. 4

4.1 리눅스 네트워킹 커널 개발의 간략한 역사. . . . 4

4.2 리눅스 네트워킹에 관한 다른 정보는 어디에서 얻을것인가? . . . 5

4.3 어디에서 리눅스에 국한되지 않은 네트워크 정보를 얻을수 있는가? . . . 6

제 5 절 일반적인 네트워크 설정과 관련되 정보. 7

5.1 시작하려면 무엇이 필요한가? . . . 7

5.1.1 현재의 커널 소스 . . . 7

5.1.2 현재의 네트워크 툴. . . . 7

5.1.3 네트워크 응용 프로그램 . . . 8

5.1.4 Addresses. . . . 8

5.2 어디에 설정 명령을 넣어야 하는가? . . . 10

5.3 네트워크 인터페이스 만들기 . . . 11

5.4 네트워크 인터페이스 설정. . . . 11

5.5 네임 해석기(Name Resolver)의 설정. . . . 12

5.5.1 이름에는 무엇이 있나? . . . 13

5.5.2 여러분에게 필요할 정보. . . . 13

5.5.3 /etc/resolv.conf . . . 14

5.5.4 /etc/host.conf . . . 14

5.5.5 /etc/hosts 14
5.6 루프백 인터페이스의 설정. 15
5.7 라우팅. 15
5.7.1 라우팅된 프로그램은 무엇을 하는가? 17
5.8 네트워크 서버와 서비스의 설정. 18
5.8.1 /etc/services 19
5.8.2 /etc/inetd.conf 24
5.9 네트워크에 관계된 다른 설정 파일들. 27
5.9.1 /etc/networks 28
5.10 네트워크 보안과 억세스 콘트롤. 28
5.10.1 /etc/ftpusers 29
5.10.2 /etc/securetty 29
5.10.3 tcpd 호스트 억세스 콘트롤 메카니즘. 29
5.10.4 /etc/hosts.equiv 31
5.10.5 ftp 데몬의 적절한 설정. 31
5.10.6 네트워크 파이어월링. 31
5.10.7 다른 제안들. 31

제 6 절 Network Technology Specific Information. 31

6.1 ARCnet 31
6.2 Appletalk(AF_APPLETALK) 32
6.2.1 Appletalk 소프트웨어의 설정. 32
6.2.2 Appletalk를 경유한 리눅스 파일시스템의 탐험. 33
6.2.3 Appletalk 와 리눅스 프린터 공유하기. 33
6.2.4 Appletalk 소프트웨어 시작하기. 33
6.2.5 appletalk 소프트웨어의 테스트. 34
6.2.6 appletalk 소프트웨어의 주의할점. 34
6.2.7 그외의 정보. 34
6.3 ATM 34
6.4 AX25 (AF_AX25) 34
6.5 DECNet. 34
6.6 EQL - multiple line traffic equaliser 35
6.7 이더넷. 35
6.8 FDDI 36
6.9 Frame Relay 36
6.10 IP 어카운팅. 40
6.11 IP 알리아싱. 41

6.12 IP 파이어월 42
6.13 IPX (AF_IPX) 44
6.14 IPv6 45

제 1 절 이전 버전과 달라진 것.

```
Additions:
          lots of things.
  Corrections/Updates:
          everything.
```

제 2 절 소개

최초의 NET-FAQ는 공식적인 리눅스 도큐먼트 프로젝트가 시작되기전 나와 Matt Welsh에 의해 리눅스 네트워킹과 관련되어 종종 나타나는 질문들에 대한 답을 하기위해서 작성되었다. 그것은 초기버전의 리눅스 네트워킹 커널에 대한 부분 까지 다루고 있다. net-2-howto는 net-faq를 이어받은 것이며, 최초의 LDP howto 문서의 하나였고, 리눅스 네트워킹 소프트웨어의 버전 2, 후에 버전 3로 불리 는 것들에 대한 내용가지도 포함하고 있다. 이 문서는 다시 net-2-howto를 이어 받은 것이며 이것은 오직 버전 3의 리눅스 네트워킹 커널과 관계가 있다.

이 문서의 초기버전의 그것의 범위에 들어가는 막대한 양의 내용들로 인해 그 크기가 무척 커졌었다. 이러한 문제를 해결하기 위해 특정한 네트워킹 토픽을 다루는 howto문서들이 생겨나게 되었다. 이 문서는 관계가 되는 부분에서 그들 문서로의 포인터를 포함하며, 다른 문서에서 다루어지지 않는 영역을 다루게 될 것이다.

2.1 Feedback

나는 feedback과 가치가 있는 투고에 대해 항상 감사한다. 어떤 feedback이나 투고라도 나에게 e-mail 로 보내주기 바란다. <*terry@perf.no.itg.telstra.com.au* `<mailto:terry@perf.no.itg.telstra.com.au>`>.

제 3 절 이 문서를 어떻게 사용할 것인가(net-3-howto-howto?).

이 문서의 포맷은 이전버전의 것과는 사뭇 다르다. 나는 여러분이 관심을 갖지 않으면 놓칠수 있는 커다란 정보를 가진 것들을 앞부분에, 나머지 부분으로 이 어지는 특정 기술 부분을 이해하기 위해서 반드시 알아두어야 할 일반적인 것들 을 다음에 볼수있도록 이 문서를 다시 그룹지어 놓았다.

Read the generic sections

이 부분은 모든, 나중에 기술될 거의 모든 부분에 적용되며, 여러분이 이 문서의 내용을 이해하기 위해서 매우 중요한 것들이다.

Consider your network

여러분은 자신의 네트워크가 어떻게 구성되어 있는가(또는 어떻게 구성 될것인가)를 알아야하며 정확히 어떤 하드웨어와 어떤 기술을 사용할 것 인가도 알고 있어야 한다.

Read the technology specific sections related to your requirements

여러분이 무엇을 원하는지를 알고있을때, 여러분은 각 컴포넌트로 관심 을 갖을수 있다. 이 부분은 특정 기술에 대한 세부적 사항만을 다룬다.

Do the configuration work

실제로 자신의 네트워크를 설정하도록 노력해야 하며, 발생하는 모든 문 제들을 신중히 기록해야 한다.

Look for further help if needed

이 문서가 해결에 도움을 주지않는 문제가 발생하였다면, 어디서 도움을 얻어야 할지, 또는 어디에 버그를 리포트해야 하는지에 관한 부분을 읽어보아라.

Have fun!

네트워킹이란 매우 재미있는 것이다. 그것을 즐겨라 !

제 4 절 General Information about Linux Networking.

4.1 리눅스 네트워킹 커널 개발의 간략한 역사.

이미 나와있는 implementation 만큼이나 잘 동작하는 완전히 새로운 tcp/ip 프 로토콜 스택의 커널 implementation을 개발한다는 것은 쉬운일이 아니었다. 이 미 나와있는 implementation의 하나를 포팅하지 않기로 하는 결정은 U.S.L에 의 해 내려진 법정 경우로 인해 제한된 카피라이트에 의해서 현존하는 implementat ion 들이 지장을 받지는 않을지 확실하지 않은경우, 또는 이미 나와있는것보다 낫게 완전히 새로운 것으로 만들기 위한 열정이 있을때에만 이루어졌다.

커널의 네트워크 코드 개발을 이끌어가기 위한 최초의 자원자는 Ross Brio였다. <*biro@yggdrasil.com* `<mailto:biro@yggdrasil.com>`>. Ross는 간단하고 불완전하나 매우 유용한 implementati on 루틴을 만들었는데 이것은 WD-8003 네트워크 인터페이스 카드의 이더넷 드라 이버로 보완되어졌다. 이것은 많은 사람들이 이 소프트웨어로 테스팅과 실험을 하기에 충분했으며, 어떤 사람들은 이 설정을 통해 자신의 머신을 가까스로 인 터넷상에 올려놓기도 하였다. 이로인해 네트워킹 지원을 개발하고 있던 리눅스 공동체내에세의 압력이 일어났고, 결국 Ross에게 있었던 불공정한 압력과 이익 에 치중하던 그의 위원회로 인하여 Ross는 리더 개발자의 자리에서 물러나게 되 었다.

논쟁만을 일삼던 환경속에서도 프로젝트를 시작하고 실제로 유용한 무언가를 내 놓기 위한 책임을 받아들이는 그의 노력은 이후의 작업에 촉매가 되었으며, 현 재의 성공에 대한 가장 본직적인 요소로 자리잡고 있다.

Orest Zborowski는 <*obz@Kodak.COM* `<mailto:obz@Kodak.COM>`> 리눅스를 위한 최초의 BSD 소켓 프로그래밍 언터페이스를 만들어냈다. 이것은 이미 나와있던 네트워크 응용프로그램들을 많 은 수정없이 리눅스에서 사용할수 있게 해줌으로써 커다란 진보를 이루어냈다.

이때쯤 다른 어딘가에서 Laurence Culhane은 <*loz@holmes.demon.co.uk* `<mailto:loz@holmes.demon.co.uk>`> 리눅스 에 SILP 프로토콜을 지원하기 위한 최초의 드라이버를 개발해 냈다. 이것은 이 더넷 네트워킹 억세스를 할수없던 많은 사람들이 새로운 네트워킹 소프트웨어를 실험할수 있도록 해주었다. 또 어떤 사람들은 이 소프트웨어를 인터넷을 접속하 는데 사용하기도 하였다. 이것은 리눅스가 충분한 네트워크 지원을 갖게된다면 실현될수있는 가능성들을 많은 사람들이 맛볼수 있게 하였으며, 기존의 네트워 킹 소프트웨어를 가지고 실제로 사용하고 실험하는 사용자수를 늘어나게 했다.

네트워킹 지원을 하는데 일익을 담당한 또 한사람은 바로 Fred Van Kempen이었 다. <*waltje@uwalt.nl.mugnet.org* `<mailto:waltje@uwalt.nl.mugnet.org>`>. Ross가 리드 개발자로서의 위치에서 물러난 불확실한 기간후에 Fred는 그의 시간과 노력을 제공하였으며 본질적으로 경쟁자 가 없는 역할을 받아들였다. Fred는 리눅스 네트워킹 소프트웨어를 이끌어가는 방향에 대한 야심찬 계획을 가지고 있었으며, 이러한 방향으로 진행해 나갔다. Fred는 'NET-2' 커널 코드라 불리는 일련의 네트워킹 코드를 만들어 냈는데(Ros s의 것에서 유래한 'NET'), 이것은 많은 사람들이 유용하게 이용할수 있었다.

Fred는 정식으로 몇개의 개혁안은 개발 예정안에 집어넣었는데, 다이나믹 디바 이스 인터페이스, 아마추어 라디오 AX.25 프로토콜 지원, 좀더 모듈적으로 디자 인된 네트워킹 implementation들이 그것들이다. Fred의

NET-2 코드는 꽤 많은 수의 열광자들에 의해 사용되었으며, 그 수는 그 소프트웨어가 동작한다는 말이 퍼짐에 따라 계속해서 늘어갔다. 이 시기의 네트워킹 소프트웨어는 스탠다드 커 널에 대한 수많은 패치에 불과했으며, 보통의 릴리즈에 포함되지도 않았다. net -faq와 이어지는 net-2-howto는 그 모든것들이 동작하도록 꽤 복잡한 프로시져 를 기술한다. Fred의 촛점은 스탠다드 네트워크 implementation에 대한 신기술 의 개발에 있었으며 이것은 시간이 걸리는 일이었다. 사용자들을 무언가 믿을만 하게 동작하고 80% 정도의 사용자를 만족시킬수 있는 것을 기대하며 인내심을 잃어갔으며 Ross의 경우와 마찬가지로 리드 개발자로서의 Fred에 대한 압력이 생겨났다.

Alan Cox <*iialan@www.linux.uk.org* `<mailto:iialan@www.linux.uk.org>`>는 이러한 상황을 타개하기 위한 해결책을 제시했다. 그는 Fred의 net-2 코드를 가져다 디버깅을 하고 안정적으로 만들어 서 인내심없는 유저들을 만족시키고, 압력으로부터 Fred를 구제해 주는동안 그 가 자신의 작업을 계속할수 있게 해주었다. Alan은 'Net-2D(ebugged)'라 불리는 자신의 첫째 버전인 리눅스 네트워킹 코드를 성공시키며 이러한 일들을 해나갔 다. 이 코드는 많은 전형적인 환경속에서 믿을만하게 동작했으며 사용자들을 행 복하게 해주었다. Alan은 확실히 그 프로젝트에 공헌할 수 있는 자신만의 아이 디어와 기술을 가지고 있었으며, NET-2 코드의 방향과 관계된 많은 논의들이 이 루어졌다. 거기에서 리눅스 네트워킹 사회속에서 서로다른 두개의 그룹이 생겨 났는데, 하나는 '일단 동작하게 만들고, 그 뒤에 더 잘 만들자'라는 철학을 가 지고 있었고, 다른 하나는 '아예 처음부터 잘 만들자'라는 철학을 가지고 있었 다. Linus가 결국에는 중재를 하고, Alan의 개발 노력에 지지를 표하며, 그의 코드를 스탠다드 커널 배포본에 포함시켰다. 이것은 Fred를 곤란한 입장으로 만 들었다. 계속되는 개발들이 그 코드를 사용하고 테스팅할 기반을 잃게 되며, 그 것은 또한 개발 진행과정이 더뎌지고 어려워짐을 의미했다. Fred는 짧은 기간동 안 개발을 계속하다 결국은 포기하고 말았으며, Alan 리눅스 네트워킹 커널 개 발 노력의 새로운 리더가 되었다.

Donald Becker <*becker@cesdis.gsfc.nasa.gov* `<mailto:becker@cesdis.gsfc.nasa.gov>`>는 곧 네트워크의 로우 레벨 국면 에 대한 자신의 재능을 드러냈으며 막대한 양의 이더넷 드라이버를 개발해냈다. 현재의 커널에 들어가는 거의 모든 이더넷 드라이버는 Donald에 의해 개발된것 이다. 중요한 공헌을 한 다른 사람들도 있지만 Donald의 업적은 정말로 풍부하 고 특별한 것이었음을 언급한다.

Alan은 얼마동안 계속해서 NET-2-Debugged 코드를 개선해 나갔으며 'TODO' 리스 트상에 어드레스되지 않은채로 남아있는 문제들을 진행해 나갔다. 리눅스 커널 1.3.*가 이빨을 드러낼때쯤까지 커널 네트워킹 코드는 현재 버전의 기반이되는 NET-3까지 발전해 갔다. Alan은 네트워킹 코드의 수많은 다른 국면에서의 작업 을 해나갔으며 리눅스 네트워킹 커뮤니티에서 온 많은 재능있는 사람들의 도움 을 받아 모든 방향에서 코드를 성장시켜 나갔다. Alan은 다이나믹 네트워크 디 바이스와 최초의 표준 AX.25 그리고 IPX implemenatation 을 만들어 냈다. Alan 은 계속해서 그 코드를 주물럭거렸고(? ^^;) 지금의 상태에 이르기까지 발전시 켜 나갔다.

PPP의 지원은 Michel Callahan <*callahan@maths.ox.ac.uk* `<mailto:callahan@maths.ox.ac.uk>`>과 Al Longyear< *longyear@netcom.com* `<mailto:longyear@netcom.com>`>에 의해서 추가되었는데 이것은 네크워킹의 사용을 위해 리눅스 를 사용하는 사람들이 증가에 대해 지나치게 비판적이었다.

Jonathon Naylor <*jsn@cs.nott.ac.uk* `<mailto:jsn@cs.nott.ac.uk>`>는 Alan의 AX.25 코드에 NetRom 프로토콜 지원을 추가함으로써 커다란 공헌을 했다. AX>25/NetRom의 지원은 대단히 중요 한데 리눅스를 제외한 다른 어떤 운영체제도 이를 표준의 지원으로 삼고있지 않 기 때문이다.

리눅스 네트워킹 소프트웨어의 발전에 중요한 공헌을 한 사람은 물론 그 외에도 많이 있다. 이들중 몇몇을 여러분은 기술적인 특정부분에 마주치게 될것이다. 그 외의 다른사람들은 모듈, 드라이버, 버그 수정, 제안, 테스트 리포트, 그리 고 정신적인 지지등에 공헌했다. 모든 것들이 다 중요한 역할을 했다고 주장할 수 있으며 그들이 할수있는 것들을 제공했다. 리눅스 커널 네트워킹 코드의 개 발은 리눅스 스타일의 부정부주의적 개발의 결과로 훌륭한 예가 된다. 그것이 여분을 놀라게하지 않았다면, 놀랄준비를 하라. 그것의 개발은 아직도 끝나지 않았다.

4.2 리눅스 네트워킹에 관한 다른 정보는 어디에서 얻을것인가?

리눅스 네트워킹에 관하여 좋은 정보를 얻을수 있는 몇가지 장소가 있다.

현재의 리눅스 커널 네트워킹 코드의 메인테이너인 Alan Cox는 리눅스 네트워킹 의 현재 그리고 새로이 개발되는 것들을 조명하는 www 페이지를 운영하고 있다. <*www.uk.linux.org/NetNews.html* `<http://www.uk.linux.org/NetNews.html>`>.

다른 좋은 장소는 Olaf Kirch가 쓴 Network Administration Guide란 책이다. 이 것은 Linux Document Project의 작업물이며 <*sunsite.unc.edu/LDP/* `<http://sunsite.unc.edu/LDP/>`> 여 러분은 이것은 Network Administration Guide HTML 버전으로 언제라도 읽을수 있다. <*sunsite.unc.edu/LDP/LDP/nag/nag.html* `<http://sunsite.unc.edu/LDP/LDP/nag/nag.html>`> 그렇지 않으면 여러분은 the sunsite.unc.edu LDP ftp archive로 다양한 포맷으로 된것을 구할수도 있 다. <*sunsite.unc.edu/pub/Linux/docs/LDP/network-guide/* `<ftp://sunsite.unc.edu/pub/Linux/docs/LDP/network-guide/>`> Olaf Kirch의 책은 이해하기가 쉽고 리눅스 상에서의 네트워크 설정에 대해 하이 레벨의 관점 을 제공한다.

리눅스 뉴스 계층에서 네트워킹과 관련된 분야에서 공헌하는 뉴스그룹도 있다. comp.os.linux.networking <news:comp.os.linux.networking>

리눅스 네트워킹에 관계된 질문을 할수있는 곳으로 등록할수 있는 메일링 리스 트도 있다. 등록을 하기 위해서는 다음의 메일을 보내라:

To: *majordomo@vger.rutgers.edu* `<mailto:majordomo@vger.rutgers.edu>`

```
        Subject: anything at all
        Message:
    subscribe linux-net
```

많은 IRC 네트워크 상에는 종종 리눅스 채널 #linux가 있으며 여기에서 사람들 은 네트워킹에 관한 질문의 답을 들을수도 있다. 어떤 문제점을 리포팅할때는 가능한 한 관련된 사항을 많이 포함해야 하는것을 기억해라. 특히 여러분은 자신이 사용하는 소프트웨어의 버전을 명시해야 하는 데 특히 커널 버전, pppd나 dip같은 툴의 버전, 그리고 여러분이 경험한 문제의 특성을 정확히 언급해야 한다. 이것은 여러분이 내린 명령과 나타난 에러메시지 의 정확한 구문을 기록해야 함을 의미한다.

4.3 어디에서 리눅스에 국한되지 않은 네트워크 정보를 얻을수 있는가?

여러분이 tcp/ip 네트워킹의 기본적인 학습을 마친상태라면 다음의 문서를 찾아 볼것을 권장한다.

tcp/ip introduction

this document comes as both a text version <*athos.rutgers.edu/runet/tcp-ip-admin.doc* `<ftp://athos.rutgers.edu/runet/tcp-ip-admin.doc>`> and a postscript version <*athos.rutgers.edu/runet/tcp-ip-admin.ps* `<ftp://athos.rutgers.edu/runet/tcp-ip-admin.ps>`>.

tcp/ip에 관한 좀더 세부적인 정보들을 알고난 뒤라면 다음의 것을 강력히 추천 한다.

"Internetworking with TCP/IP"

```
        by Douglas E. Comer
    ISBN 0-13-474321-0
        Prentice Hall publications.
```

유닉스 호환 환경에서 네트워크 어플리케이션 작성에 관해 배우고 싶다면, 다음 을 강력히 추천한다.

"Internetworking with TCP/IP"

```
        by Douglas E. Comer
    ISBN 0-13-474321-0
        Prentice Hall publications.
```

comp.protocols.tcp-ip를 뒤져볼수도 있다. <news:comp.protocols.tcp-ip>

인터넷과 tcp-ip 프로토콜 suite과 관련된 특정 기술 정보의 중요한 소스하나가 바로 RFCs이다. RFC는 'Request For Comment'의 머릿글자이며, 인터넷 프로코톨 의 표준을 제안하고 도큐먼팅 하는 표준적인 방법이다. RFC를 보관하고 있는곳 은 많이 있다. 이 사이트의 상당부분은 FTP 사이트이며 특정 키워드로 RFC 데이 터베이스를 검색할수 있게 연결시켜논 www 사이트도 있다.

RFC를 위한 소스중 하나가 다음에 있다. <*pubweb.nexor.co.uk/public/rfc/index/rfc.html* `<http://pubweb.nexor.co.uk/public/rfc/index/rfc.html>`>

제 5 절 일반적인 네트워크 설정과 관련되 정보.

다음의 부분들을 여러분이 실제로 자신의 네트워크를 설정하기전에 알고 이해할 필요가 있는 것들이다. 이들은 여러분이 설치하려는 네트워크의 정확한 특성에 관계없이 적용되는 기본적인 원리들이다.

5.1 시작하려면 무엇이 필요한가?

네트워크를 설치하고 설정하기 전에 필요한 것이 몇가지 있다. 가장 중요한것이 다음의 것들이다.

5.1.1 현재의 커널 소스

현재 여러분이 운영하고 있는 커널이 여러분이 사용하고 싶어하는 네트워크 타 입이나 카드등을 지원하지 않을수도 있기때문에 적절한 옵션으로 커널을 다시 컴파일하기 위해서 커널 소스가 필요하다.

여러분은 ftp.funet.fi에서 최신버전의 커널 소스를 얻을수 있다. <*tp.funet.fi/pub/Linux/PEOPLE/Linus/v2.0* `<ftp://ftp.funet.fi/pub/Linux/PEOPLE/Linus/v2.0>`>

보통 커널 소소는 /usr/src/linux 디렉토리로 tar이미지가 풀어져야 한다. 어 떻게 패치를 적용하고 커널을 만들것인가에 대한 정보를 알기 위해서는 kernel- howto를 읽어야 한다. kernel-module의 설정에 대해 알고 싶다면 module-howto를 읽어야 한다.

특별히 언급하지 않는 한 여러분은 표준의 커널 릴리즈(버전 넘버의 두번재 dig it이 짝수로 된것)를 고집하기 바란다. 개발 릴리즈 커널은(두번째 digit이 홀 수인 것) 시스템상의 다른 소프트웨어와 문제를 일으킬 수 있는 구조 자체의 또 는 다른 변화가 있을수 있다. 여러분이 그러한 종류의 문제를 해결할수 있다고 확신하지 않는한, 또 잠재적인 소프트웨어의 에러를 처리가 확실하지 않은 한 그것을 사용하지 말라.

5.1.2 현재의 네트워크 툴.

네트워크 툴들은 리눅스의 네트워크 디바이스들을 설정하기 위해 사용하는 프로 그램들이다. 예를들어 이런 툴들은 디바이스에 주소를 할당하게 해주거나 라우 트를 설정할수 있게 해준다.

대부분 최신의 리눅스 배포본은 네트워크 툴들과 함께 제공되므로, 여러분이 배 포본으로 부터 인스톨을하고, 네트워크 툴을 인스톨하지 않았다면 이것을 해야 만 한다.

배포본으로 설치하지 않았다면 여러분은 직접 소스를 구해다가 컴파일해야 할 필요가 있다. 이것은 그다지 어렵지 않다.

네트워크 툴은 현재 Bernd Eckenfels에 의해 관리되며, ftp.inka.de에서 구할 수 있다. <*ftp.inka.de/pub/comp/Linux/networking/NetTools/* `<ftp://ftp.inka.de/pub/comp/Linux/networking/NetTools/>`> and are 또 다음에 미러링되어 있다. <*ftp.linux.uk.org/pub/linux/Networking/PROGRAMS/NetTools/* `<ftp://ftp.linux.uk.org/pub/linux/Networking/PROGRAMS/NetTools/>`>. 여러분이 사용하고자 하는 커널 버전과 가장 적합한 버전을 선택해야 함을 명심 하라. 그리고 인스톨을 하려면 패키지 안의 지시를 따르라.

이글을 작성하는 때의 현재버전을 인스톨하고 설정하고자 한다면 다음과 같이 해야한다.

```
#
  # cd /usr/src
  # tar xvfz net-tools-1.32-alpha.tar.gz
  # cd net-tools-1.32-alpha
  # make config
  # make
  # make install
  #
```

추가로, 파이어 월을 설정하고, IP 머스커레이드를 사용하고 싶다면 ipfwadm 명 령이 있어야 한다. 그것의 최근버전을 *ftp.xos.nl* `<ftp://ftp.xos.nl>`에서 구할수있다. <*ftp.xos.nl/pub/linux/ipfwadm* `<ftp://ftp.xos.nl/pub/linux/ipfwadm>`>.또, 그곳에서 구할수있는 버전은 여러가 지가 있으므로, 여러분의 커널과 버전과 가장 가까운 것을 선택하도록 해야한 다.

이 글을 작성하는 시기의 현재버전을 설치하고 설정하려면, 다음과 같이 하라.

```
#
  # cd /usr/src
  # tar xvfz ipfwadm-2.3.0.tar.gz
  # cd ipfwadm-2.3.0
  # make
  # make install
  #
```

5.1.3 네트워크 응용 프로그램

네트워크 응용 프로그램이란 telnet, ftp 등의 프로그램과 각각의 서버 프로그 램을 말한다. David Holland<*dholland@hcs.harvard.edu* `<mailto:dholland@hcs.harvard.edu>`>는 이제 이들 프로그램 의 대부분을 관리하고 있다. 여러분은 *ftp.linux.uk.org* `<ftp://ftp.linux.uk.org>`에서 그것을 구할수가 있다. <*ftp.uk.linux.org/pub/linux/Networking/base* `<ftp://ftp.uk.linux.org/pub/linux/Networking/base>`>. 이 글을 작성하는 시기의 현재버전을 설치, 설정하려면 다음과 같이 하라.

```
#
  # cd /usr/src
  # tar xvfz /pub/net/NetKit-B-0.08.tar.gz
  # cd NetKit-B-0.08
  # more README
  # vi MCONFIG
  # make
  # make install
  #
```

5.1.4 Addresses.

인터넷의 프로토콜 어드레스는 4 바이트로 이루어진다. 이것에 대한 합의는 주 소를 'dotted decimal notation'이라 불리는 꼴로 기록하는 것이다. 이러한 for m에서 각각의 바이트는 0이 아닌 이상에는 앞의 0을 빼버린 (0 255) 10진수로 변환되며, 각각의 바이트는 '.' 문자로 분리되어 쓰여진다. 그것의 편리함으로 인해 호스트와 라우터의 각각의 인터페이스는 IP 어드레스를 가진다. 몇몇 환경 속에서는 단일 머신상의 각각

의 포트에 대하여 같은 IP 어드레스를 사용하는 것 이 합법적이지만 보통 각 인터페이스는 자신만의 주소를 갖기마련이다.

인터넷 프로토콜 네트워크는 인접해 있는 IP 어드레스들의 시퀀스이다. 네트워 크상의 모든 어드레스는 공통적으로 몇개의 점(digit)을 어드레스내에 포함하고 있다. 한 네트워크의 어드레스에서 공통적인 부분은 'network portion'이라 불 린다. 'network portion'을 제외한 나머지 부분은 'host portion'이라고 불린 다. 네트워크의 모든 어드레스에 의해 공유되는 bit은 netmask라 불리는데, 어 떤 어드레스가 그것이 적용된 네트워크에 속하는지, 속하지 않는지를 결정짓는 것이 netmask의 역할이다. 다음의 예를 살펴보자.

```
----------------  ---------------
  Host Address       192.168.110.23
  Network Mask       255.255.255.0
  Network Portion    192.168.110.
  Host portion                  .23
  ----------------   ---------------
  Network Address    192.168.110.0
  Broadcast Address  192.168.110.255
  ----------------   ---------------
```

netmask내에서 'bitwise anded'된 어드레스는 자신이 속한 네트워크의 어드레스 를 공개하게 될것이다. 그러므로 네트워크의 어드레스 범위중에서 가장 낮은 넘 버의 어드레스는 항상 network address가 되는것이다. 또한 항상 0으로 코드된 'host portion'을 갖게 된다.

브로드캐스트 어드레스는 자신의 독특한 어드레스에 더하여 네트워크내의 모든 호스트가 듣는 특별한 어드레스이다. 이 어드레스는 네트워크 상의 모든 호스트 가 동시에 듣게끔 할때 데이터그램이 보내지는 어드레스이다. '브로드캐스트란 어떤것이어야 하는가'를 위해 사용되는 공통적으로 사용되는 두가지의 표준사항 이 있다. 가장 널리 받아들여지고 있는것은 해당 네트워크에서 가장 높은 값을 지니는 어드레스를 브로드캐스트 어드레스로 사용하는 것이다. 위의 예에서 이 것은 192.168.110.255가 된다. 다른 사이트에서는 몇가지 이유로 네트워크 어드 레스를 브로드캐스트 어드레스로 사용하는 방법을 채택했다. 실제로 어느것을 사용하는가 하는것은 별로 문제가 되지 않지만 네트워크상의 모든 호스트가 같 은 브로드캐스트 어드레스로 설정되어 있는가는 반드시 확인해야 한다. IP 프로 토콜 개발 초기의 언젠가 관리상의 이유로 임의의 그룹의 어드레스가 네트워크 에 형성되었고, 이러한 네트워크는 'class'라 불리는 것으로 그룹화되었다. 이 클래스는 할당할수 있는 표준 사이즈 네트워크의 갯수를 제공해준다. 할당된 범위는 다음과 같다.

```
----------------------------------------------------------
  | Network | Netmask       | Network Addresses             |
  | Class   |               |                               |
  -----------------------------------------------------------
  |    A    | 255.0.0.0     | 0.0.0.0     - 127.255.255.255 |
  |    B    | 255.255.0.0   | 128.0.0.0   - 191.255.255.255 |
  |    C    | 255.255.255.0 | 192.0.0.0   - 223.255.255.255 |
  |Multicast| 240.0.0.0     | 224.0.0.0   - 239.255.255.255 |
  -----------------------------------------------------------
```

어떤 어드레스를 사용해야 하는가는 정확히 여러분이 해야할일이 무엇인가에 따 라 달라진다. 필요한 모든 어드레스를 얻기 위해서는 다음을 조합해서 사용해야 한다.

기존의 IP 네트워크상에 리눅스 머신을 설치한다. IP 네트워크에 리눅스 머신을 설치하고자 한다면 네트워크 관리자와 접촉하여 다음의 정보를 물어봐야 한다.

- Host IP Address

- IP network address
- IP broadcast address
- IP netmask
- Router address
- Domain Name Server Address

그런뒤 리눅스 네트워크 디바이스를 이 세부사항으로 설정해 한다. 이것을 하지않고는 여러분이 한 설정이 동작하리라 기대할수는 없다.

인터넷에 연결되지 않는 완전히 새로운 네트워크 만들기 인터넷에 연결되지 않을 사적인 네트워크를 만든다면 여러분은 아무 어드레스나 선택할수 있다. 그러나 안전성과 일관성의 이유로 특별히 예약되어있는 IP 어드레스가 있다. 이것은 다음과 같이 RFC1597에 명시되어 있다.

```
   -----------------------------------------------------------
      |         RESERVED PRIVATE NETWORK ALLOCATIONS           |
      -----------------------------------------------------------
      | Network | Netmask        | Network Addresses           |
      | Class   |                |                             |
      -----------------------------------------------------------
      |    A    | 255.0.0.0      | 10.0.0.0    - 10.255.255.255  |
      |    B    | 255.255.0.0    | 172.16.0.0  - 172.31.255.255  |
      |    C    | 255.255.255.0  | 192.168.0.0 - 192.168.255.255 |
      -----------------------------------------------------------
```

우선 네트워크의 크기를 결정한뒤 원하는 만큼의 어드레스를 선택해야 한다.

5.2 어디에 설정 명령을 넣어야 하는가?

리눅스 시스템 부트 프로시져로의 서로다든 접근법이 몇가지 있다. 커널이 부팅 된 뒤에 그것은 항상 'init'이라는 프로그램을 실행한다. init은 /etc/inittab 이라는 설정파일을 읽고 부트 과정을 시작한다. init에는 몇개의 다른 특질(fla vours)이 있으며, 이러한 다양성은 리눅스 배포본과 머신의 다양성의 가장 큰 원인이 된다.

흔히 /etc/inittab 파일은 다음과 같은 엔트리를 포함하고 있다.

```
    si::sysinit:/etc/init.d/boot
```

이 라인은 실제로 부트 시퀀스를 관리하는 shell 스크립트의 이름을 명시한다. 이 파일은 MS-DOS상의 autoexec.bat 파일과 다소 유사하다.

이 부트 스크립트에 의해 호출되는 다른 스크립트가 있으며 네트워크는 이들안 에서 설정된다.

다음 테이블이 여러분의 시스템의 가이드로 사용될지도 모르겠다.

```
-----------------------------------------------------------------------
 |Interface Config/Routing                   |Server Initialisation
 -----------------------------------------------------------------------
 |/etc/init.d/network                        |/etc/init.d/netbase
 |                                           |/etc/init.d/netstd_init
```

```
|                                            |/etc/init.d/netstd_nfs
|                                            |/etc/init.d/netstd_misc
---------------------------------------------------------------------
|/etc/rc.d/rc.inet1                          |/etc/rc.d/rc.inet2
---------------------------------------------------------------------
|/etc/sysconfig/network-scripts/ifup-<ifname>|/etc/rc.d/init.d/network
---------------------------------------------------------------------
                         <순서대로 Debian, Slackware, RedHat>
```

대개의 배포본은 많은 종류의 흔히 사용되는 네트워크 인터페이스를 설정할수 있는 프로그램을 포함하고 있다. 이것이 있다면 수동 설정을 시도하기 전에 이 것이 원하는 것을 해주는지 알아봐야 한다.

```
       ---------------------------------------
          Distrib   | Network configuration program
          ---------------------------------------
          RedHat    | /sbin/netcfg
          Slackware | /sbin/netconfig
          ---------------------------------------
```

5.3 네트워크 인터페이스 만들기

많은 유닉스 시스템에서 네트워크 디바이스는 /dev 디렉토리에서 보인다. 그러 나 리눅스에서는 그렇지 않다. 리눅스의 네트워크 디바이스는 소프트웨어 내에 서 동적으로 생성되므로 디바이스 파일이 존재할 필요가 없다.

대개의 경우 네트워크 디바이스는 초기화하는 동안 디바이스 드라이버에 의해 자동으로 만들어지고 하드웨어를 설치한다. 예를들어 이더넷 디바이스 드라이버 는 순차적으로 eth[0...n] 인터페이스를 만들고 이더넷 하드웨어를 위치시킨다. 첫번째로 발견되는 이더넷 카드는 eth0가 두번째의 이더넷 카드는 eth1이 된다.

그러나 몇몇의 경우, slip 또는 ppp와 같이, 네트워크 디바이스는 유저 프로그 램의 동작에의해 만들어진다. 디바이스에 순차적으로 이름을 붙이는것은 적용되 지만 디바이스가 부팅시에 자동적으로 만들어지지는 않는다. 이러한 이유는 이 더넷 드라이버와 달리 활성화된 slip, ppp 디바이스의 개수는 머신의 작동시간 동안 매우 다양하게 달라지기 때문이다. 이러한 경우는 다음 부분에서 자세히 다루어질 것이다.

5.4 네트워크 인터페이스 설정.

필요한 모든 프로그램과 네트워크 어드레스, 네트워크 정보를 모두 가지고 있으 면 이제 네트워크 인터페이스를 설정할 수 있다. 네트워크 인터페이스의 설정에 관해 말할때 우리는 네트워크 디바이스로의 적절한 주소 할당과 다른 환경 설정 값의 적절한 세팅에 관해 이야기를 하는것이다. 이를 위해 가장 널리 사용되는 명령은 ifconfig(interface configure)이다.

전형적으로 아래와 유사한 명령을 사용할 것이다.

```
# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
```

이 경우는 'eth0' 이더넷 인터페이스를 IP 어드레스 '192.168.0.1'로 netmask를 255.255.255.0으로 설정하는 것이다. 명령어 끝부분의 'up'은 인터페이스가 활 성화 될것을 말하고 있다.

커널은 인터페이스를 설정할때 몇가지 디폴트를 가정한다. 예를 들어 네트워크 어드레스와 브로드캐스트 어드레스를 명시할 수도 있지만, 위의 예처럼 하지 않 는다면 커널은 설정되 IP 어드레스의 클래스에 근거해 합리적으로 추측해 낼것 이다. 위의 예에서 커널은 인터페이스에 클래스-C의 네트워크가 설정된다고 가정하고 네트워크 어드레스를 '192.168.0.0'으로, 브로드캐스트 어드레스를 '192.168.0.255'로 설정한다.

ifconfig 명령에는 많은 다른 옵션이 있다. 가장 중요한 것은 다음의 것이다.

up

인터페이스를 활성화시킨다.

down

인터페이스를 비 활성화시킨다.

-arp

이 인터페이스 상에서 address resolution protocol을 사용가능하게, 사 용불가능하게 한다.

-allmulti

인터페이스 상에서 promiscuous 모드를 가능/불가능하게 한다. Promoscu ous 모드란 해당 디바이스를 향하지 않은 패킷이라도 받아들일수 있도록 하는 모드를 말한다. 이것은 tcpdump나 다른 패킷 스푸너 프로그램에서 매우 중요하다.

mtu N

이 디바이스의 MTU를 셋팅할수 있게 해준다.

netmask addr

이 디바이스가 속하는 네트워크의 netmask를 설정하게 해준다.

irq addr

이 파라메터는 특정 타입의 하드웨어에서만 동작하나, 이 디바이스의 하 드웨어 IRQ를 셋팅할수 있게 해준다.

-broadcast addr

이 파라메터는 브로드캐스트 어드레스를 향한 데이터그램의 수용을 가능 하게 세팅하거나, 혹은 이 데이터그램의 불가능하게 한다.

-pointopoint addr

이 파라메터는 slip, ppp등의 point to point 링크의 반대끝에 있는 머 신의 주소를 설정하도록 해준다.

hw <type> <addr>

이 파라메터는 네트워크 디바이스의 특정 타입의 하드웨어 어드레스를 설정하도록 해준다. 이더넷에는 그다지 유용하지 않으나 AX.25와 같은 다른 타입의 네트워크에서는 유용하다.

ifconfig

명령은 어느 네트워크 인터페이스에서나 사용할수 있을것이다. pppd, dip 같은 유저 프로그램들은 네트워크 디바이스를 만들때 자동으로 설정 하므로 수동의 ifconfig 명령사용은 필요하지 않다.

5.5 네임 해석기(Name Resolver)의 설정.

’Name Resolver’는 표준 리눅스 라이브러리의 일부이다. 이것의 가장 큰 기능은 ’ftp.funet.fi’ 같이 인간에게 친숙한 호스트네임을 128.214.248.6 처럼 머신에 게 친숙한 IP 어드레스로 변환하는 서비스를 제공하는 것이다.

5.5.1 이름에는 무엇이 있나?

여러분은 인터넷 호스트 네임의 모습에는 익숙하나, 그것이 어떻게 구성되는지 는 이해하지 못할지도 모른다. 인터넷 도메인 네임은 본질적으로 계층구조를, 다시 말해서 트리 구조를 갖는다. 'domain'이라고 하는 것은 가족의 또는 그룹의 이름이다. 'domain'은 'subdomain'으로 나뉘어 질수 있다. 'toplevel' 도메인은 서브도메인이 아닌 도메인을 의미한다. Top Level Domain은 RFC920에 명시되어 있다. 다음은 가장 흔한 몇가지 TOP LEVEL 도메인의 예이다.

COM

Commercial Organisations(상업기관)

EDU

Educational Organisations(교육기관)

GOV

Government Organisations(정부기관)

MIL

Millitary Organisations(군사기관)

ORG

Other organisations(다른기관)

Country Designator

these are two letters codes that represent a particular country.

(특정 국가를 나타내기 위한 두글자 코드. 예: 대한민국 -> kr) 각각의 top level 도메인인 서브도메인을 가지고 있다. 국가의 이름에 기반한 top level 도메인은 com, edu, gov, mil, org 도메인에 기반한 서브도메인으로 사용된다. 예로써, 오스트레일리아의 상업기관 또는 정부기관을 나타내기 위해 서는 com.au, gov.au 등으로 끝낸다. '.us'라는 국가코드를 가지고 있음에도 불 구하고, 국가 기반의 top level 도메인을 갖지 않는 기관들은 역사적인 이유로 미국내의 기관은 나타내기 위해 사용된다.

다음 레벨의 분할은 기관의 이름을 나타내기 위해서 사용된다. 좀더 깊은 서브 도메인은 본질적으로 매우 다양한데, 다음 레벨의 서브 도메인은 종종 기관의 부문별 구조에 근거하게 된다. 그러나 이것은 기관의 네트워크 관리자에 의해 합리적으로 의미있게 고려된 기준에 기반을 두게 된다.

네임의 가장 왼쪽 부분은 항상 호스트 머신에 할당된 독특한 이름이며 '호스트 네임'이라 불린다. 네임에서 호스트네임의 오른쪽 부분은 '도메인 네임'이라 불 리며 이것의 풀네임은 'Fully Qualified Domain Name'이다.

내 이메일 호스트를 한 예로 들면, 'Fully Qualified Domain Name'은 'perf.no. itg.telstra.com.au'이다. 이것은 호스트네임이 'perf' 도메인네임이 'no.itg.t elstra.com.au'임을 의미한다. 도메인 네임은 나의 국가 Australia에 근거한 to p level 도메인에 기반을 두며 내 이메일 어드레스는 상업기관에 속하므로 다음 레벨 도메인으로 '.com'을 가지고 있다. 회사의 이름은 'Telstra'이며 내부의 작명구조는 조직구조에 기반을 두는데, 이 경우 내 머신은 Infomartion Technol ogy Group의 Network Operation 부서에 속한다.

5.5.2 여러분에게 필요할 정보.

자신의 호스트가 어느 도메인에 속하게 되는지를 알아야 할 필요가 있다. Name resolver 소프트웨어는 'Domain Name Server'로의 요처을 만들어 냄으로써 이 네임 번역 서비스를 제공한다. 그러므로 자신이 사용할수 있는 로컬 네임서버의 IP 어드레스를 알아야 할 필요가 있다.

수정할 필요가 있는 파일이 3개 있는데, 이것을 차례로 다루어 나가겠다.

5.5.3 /etc/resolv.conf

/etc/resolv.conf는 네임 해석 코드를 위한 메인 설정 파일이다. 이것의 포맷은 매우 간단하다. 하나의 라인에 하나의 키워드를 갖는 텍스트 파일이다. 전형적 으로 사용되는 키워드는 3개가 있는데 다음과 같다.

domain

이 키워드는 로컬 도메인을 명시한다.

search

이 키워드는 호스트네임을 찾기위해 선택할수 있는 도메인네임의 리스트 를 명시한다.

nameserver

이 키워드는 많이 쓰일텐데, 네임 해석을 할때 질의를 할 도메인 네임 서버의 IP 어드레스를 명시하기 위해 사용된다.

/etc/resolv.conf 의 예는 다음 처럼 보인다.

```
domain maths.wu.edu.au
   search maths.wu.edu.au wu.edu.au
   nameserver 192.168.10.1
   nameserver 192.168.12.1
```

이 예는 분류되지 않은 네임(도메인네임을 쓰지 않은 호스트네임)에 덧붙일 디 폴트 도메인네임이 maths.wu.edu.au임을 명시하고, 그 호스트가 이 도메인에서 발견되지 않을경우 wu.edu.au 도메인에서 직접 찾아볼것을 명시한다. 두개의 네 임서버 엔트리도 제공되는데, 이들 각각은 네임을 해석할때 사용한다.

5.5.4 /etc/host.conf

/etc/host.conf 파일은 name resolver 코드의 행동을 통제하는 아이템을 설정하 는 파일이다. 이 파일의 포맷은 'resolv+' 맨 페이지에 자세히 기술되어 있다. 거의 모든 환경에서 다음의 예는 잘 동작할것이다.

```
order hosts,bind
   multi on
```

이 설정은 네임 해석기에게, 네임서버에 질의하기 전에 /etc/hosts를 체크할 것 과 /etc/hosts 파일에서 발견되는 첫번째 어드레스 뿐만아니라 모든 유효 어드 레스를 리턴할것을 말해준다.

5.5.5 /etc/hosts

/etc/hosts 파일은 로컬 호스트의 네임과 IP 어드레스를 집어넣는 곳이다. 이 파일에 호스트를 적어넣으면 IP 어드레스를 얻기 위해 도메인 네임 서버에 질의 할 필요가 없다. 이 파일을 유지하는데 불편한 점은 호스트의 IP 어드레스가 바 뀔때 자신이 업데이트를 해야한다는 점이다. 잘 관리되는 시스템의 이 파일에서 보이는 호스트네임은 루프백 인터페이스와 로컬 호스트의 네임을 위한 엔트리 뿐이다.

```
# /etc/hosts
   127.0.0.1      localhost loopback
   192.168.0.1    this.host.name
```

첫번째 엔트리에서 보여주는 바와같이 한 라인에 한개 이상의 호스트 네임을 명 시할 수도 있다. 위의 첫번째 엔트리는 루프백 인터페이스를 위한 표준 엔트리 이다.

5.6 루프백 인터페이스의 설정.

’loopback’ 인터페이스는 자신에게 접속할수 있도록 해주는 특별한 형태의 인터 페이스이다. 이를 사용하고 싶은 이유는 매우 다양한데, 네트워크의 다른 사용 자를 방해하지 않고 네트워크 소프트웨어를 테스트하고 싶을 때 등..이 좋은 예 가 된다. 약속에 의해 IP 어드레스 ’127.0.0.1’은 루프백을 위해 명시적으로 할 당되어 있다. 그러므로 어떤 머신을 운영하더라도, 127.0.0.1의 텔넷 커넥션을 열면 로컬 호스트에 도달하게 된다.

루프백 인터페이스를 설정하는것은 간단하며, 다음은 분명히 해주어야 한다.

```
# ifconfig lo 127.0.0.1
   # route add -host 127.0.0.1 lo
```

다음 부분에서 우리는 route 명령에 대해 자세히 다룰것이다.

5.7 라우팅.

라우팅은 거대한 토픽이다. 이것에 관해서는 엄청난 양의 텍스트를 작성하는 것 이 가능하다. 대부분의 사람은 라우팅에 관해 간단한 요구만을 할것이나, 몇몇 사람은 그렇지 않다. 나는 라우팅의 기초적인 기본사항만을 다룰것이다. 좀더 세부적인 정보에 관심이 있다면 이 문서의 시작부분에 있는 레퍼런스를 참고하기 바란다.

정의와 함께 시작하자. IP 라우팅이란 대체 무엇인가? 여기 내가 사용하는 정의 중 하나가 있다.

IP 라우팅이란 호스트가 받은 멀티플 네트워크 데이터그램에 의한 프로 세스이다.

예를 들어 설명하는게 좋겠다. 전형적인 오피스 라우터를 상상해보라. 그것은 아마 인터넷에 연결되지 않은 ppp 링크, 워크스테이션에 공급하는 몇개의 이더 넷 세그먼트 다른 오피스로의 ppp 링크등을 가지고 있을것이다. 어느 네트워크 커넥션으로부터 라우터가 데이터그램을 받을때, 라우팅이란 다음에 데이터그램을 어느 포트로 보낼것인가를 결정짓는 메커니즘이다. 간단한 호스트에서도 라 우팅은 필요하며 모든 인터넷 호스트는 두개의 네트워크 디바이스를 갖는데 하 나는 위에서 언급한 루프백 인터페이스이며 다른 하나는 이더넷 혹은 PPP, SLIP 등의 나머지 네트워크에 이야기를 하기위해 사용하는 디바이스이다.

OK, 그러면 라우팅이란 어떻게 동작하는 것인가? 각 호스트는 라우팅 테이블이 라 불리는 라우팅 룰의 특별한 리스트를 가지고 있다. 이 테이블은 최소 3개의 필드를 가지는 행을 포함하는데, 첫째 필드는 목적지 어드레스, 두번째 필드는 데이터그램이 라우팅될 인터페이스의 이름을, 세번째는 옵션으로 네트워크를 통해 다음 스텝으로 데이터그램을 운반할 다른 머신의 어드레를 가지고 있다. 다 음 명령을 이용하여 리눅스의 라우팅 테이블을 살펴볼수 있다.

```
# cat /porc/net/route
```

라우팅 프로세스는 매우 간단하다: 들어오는 데이터그램은 받고, 목적지 어드레 스를 조사하여 테이블상의 각 엔트리와 비교한다. 해당 어드레스와 가장 잘 부 합하는 엔트리를 선택하고 명시된 인터페이스로 데이터그램을 포워딩한다. 게이 트워에 필드가 채워져 있다면 데이터그램은 인터페이스를 경유해 그 호스트로 포워딩되며, 그렇지 않으면 인터페이스가 지원하는 네트워크에 있다고 가정한 다.

라우팅 테이블을 조작하기 위해서는 특별한 명령을 사용한다. 이 명령은 커맨드 라인 변수를 받아서 커널이 테이블내의 엔트리를 추가, 삭제, 수정하도록 요청 하는 커널 시스템 콜로 변환한다. 이 명령은 ’route’라 불린다.

간단한 예가 있다. 여러분이 이더넷 네트워크를 가지고 있다고 생각해보자. 그 것이 192.168.1.0의 어드레스를 가진 class-C의 네트워크라고 들었다. 또 자신 이 사용하기 위해 192.168.1.10을 지원하고, 192.168.1.1이 인터넷에 연결된 라우터라고 듣는다.

첫번째로 할일은 위에서 설명한대로 인터페이스를 설정하는 일이다. 다음과 같 은 명령을 사용한다.

```
# ifconfig eht0 192.168.1.10 netmask 255.255.255.0 up
```

다음에는 192.168.1.*에 매치되는 모든 어드레스를 향한 데이터그램이 이 이더 넷 디바이스로 보내지도록 라우팅 테이블에 엔트리를 추가해야 한다.

```
# route add -net 192.168.0.0 netmask 255.255.255.0 eth0
```

'-net' 변수의 사용은 이 엔트리가 네트워크 엔트리임을 route 프로그램에 알려 주기 위해서이다. 여기서 할 수있는 다른 선택은 '호스트' 라우트인데 이것은 하 나의 IP 어드레스로 명시되는 route이다.

이 라우트는 여러분의 이더넷 세그먼트상에 있는 모든 호스트와의 커넥션 성립 을 가능하게 해준다. 하지만 자신의 이더넷 세그먼트에 있지 않은 다른 모든 IP 호스트는 어떻게 되는가?

모든 가능한 목적지 네트워크의 route를 추가하는것은 대단히 어려운 일이므로, 이 작업을 간단하게 해주는 특별한 트릭이 있다. 이 트릭은 '디폴트' 라우트라 고 불린다. 디폴트 라우트는 모든 가능한 목적지와 매치되지만 빈약하므로 요청 받은 어드레스와 매치하는 어드레스가 있다면 디폴트 라우트 대신에 이것이 사용된다. 디폴트 라우트의 아이디어는 "모든것은 이곳으로"라고 하는것을 가능하 게 해준다. 이 예에서는 다음과 같은 엔트리를 사용하도록 하였다.

```
# route add default gw 192.168.1.1 eth0
```

'gw' 변수는 다음 변수가 게이트웨이(혹은 라우터 머신)의 IP 주소, 또는 네임 임을 route 명령에게 알려준다. 이 엔트리에 매치하는 모든 데이터그램은 더 먼 라우팅을 위해 그 게이트웨이로 보내져야 한다.

그래서, 완성된 설정은 다음과 같이 보일것이다.

```
# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
   # route add -net 192.168.0.0 netmask 255.255.255.0 eth0
   # route add default gw 192.168.1.1 eth0
```

자신의 네트워크 'rc' 파일을 잘 살펴본다면 이와 유사하게 보이는것을 적어도 하나는 발견하게 될것이다. 이것은 매우 일반적인 설정이다.

이제 약간 더 복잡한 라우팅 설정을 살펴보자. 인터넷으로의 PPP 링크와 사무실 의 워크스테이션에 랜 세그먼트를 공급하는 라우터를 설정한다고 상상해보자. P PP 링크 하나와 이더넷 세그먼트 3개를 가지는 라우터를 설정한다고 생각해보 자. 라우팅 설정은 다음과 같이 보일것이다.

```
# route add 192.168.1.0 netmask 255.255.255.0 eth0
   # route add 192.168.2.0 netmask 255.255.255.0 eht1
   # route add 192.168.3.0 netmask 255.255.255.0 eth2
   # route add default ppp0
```

각 워크스테이션은 위에 나타난것보다 간단한 것을 사용하겠지만, 라우터는 각 네트워크 루트(route)를 명시할 필요가 있다. 워크스테이션에 있어서 디폴트 메 커니즘은 라우터가 적절히 분리해야 하는것을 모두 잡아낼것이기 때문이다. 여러분은 위에 나타난 디폴트 라우트가 'gw'를 명시하지 않은 이유를 궁금해 할 지도 모른다. 이유는 매우 간단한데 PPP나 SLIP 같은 시리얼 링크 프로토콜은 네트워크 양단에 한대씩, 단 두대의 호스트밖에 없기 때문이다. 링크의 반대끝 에 있는 호스트를 게이트웨이로 명시하는 것은 무의미하고 과잉적인 것이다. 왜 냐면 다른 선택이 있을수 없기 때문에 이런 타입의 네트워크 커넥션에서는 게이 트웨이를 명시해줄 필요가 없다. 이더넷, 아크넷(arcnet), 토큰 링에서는 게이 트웨이를 명시해줘야 하는데 이들 네트워크에는 많은수의 호스트가 있기 때문이 다.

5.7.1 라우팅된 프로그램은 무엇을 하는가?

위에서 언급된 라우팅 설정은 목적지로 가는길이 단 하나뿐인 간단한 네트워크 어레인지먼트에서 가장 잘 적용된다. 좀더 복잡한 네트워크에서 해야할 일은 좀 더 복잡하다. 다행이도 대부분의 사람들에게 이것은 화젯거리가 되지않는다.

'수동 라우팅'이나 '정적 라우팅'의 가장 큰 문제점은 언급한 바와 같이, 네트 워크의 머신이나 링크가 망가져서 데이터그램을 다른 길로 보내야 한다면(다른 길이 있을경우) 직접 이들을 조정하고 적절한 명령을 실행해야 한다는 점이다. 당연히 이것은 느리고, 꼴사납고, 비실용적이고, 바람직하지 않은 일이다. 네트 워크에 문제가 생겼을 경우 가능한 루트를 자동으로 적용시키는 다양한 테크닉 들이 개발되었는데, 이러한 모든 그룹은 '다이나믹 라우팅 프로토콜'이라는 용 어로 느슨하게 묶여있다.

여러분은 일반적인 다이나믹 라우팅 프로토콜에 관해 들었을지도 모른다. 가장 널리 알려진 것은 아마도 RIP(Routing Information Protocol)과 OSPF(Open Shor test Path First Protocol)일 것이다. RIP는 중간 정도 사이즈 단체의 네트워크 나 빌딩 네트워크 등에서 흔히 사용된다. OSPF는 좀더 현대적이고 보다 큰 네트 워크를 다루는 설정능력이 있다. 또한 네트워크를 통해 좀저 많은 수의 길(pat h)가 있는 환경에서 적합하다. 이 프로토콜들이 일반적인 도구는 'routed'-RIP, 'gated'-RIP,OSPF 등이다. 'routed' 프로그램은 보통 배포본과 함께 재공되며, 위에서 언급한 'NetKit' 패키지에도 포함되어 있다.

다이나믹 라우팅 프로토콜을 어디서 어떻게 사용할 것인가 하는 예는, 다음처럼 보일것이다.

```
    192.168.1.0 /                         192.168.2.0 /
       255.255.255.0                         255.255.255.0
     -                                     -
     |                                     |
     |   /-----\                 /-----\   |
     |   |     |ppp0   //    ppp0|     |   |
eth0 |---|  A  |------//---------|  B  |---| eth0
     |   |     |     //          |     |   |
     |   \-----/                 \-----/   |
     |      \ ppp1           ppp1 /        |
     -       \                     /       -
              \                   /
               \                 /
                \               /
                 \             /
                  \           /
                   \         /
                    \       /
                     \     /
                  ppp0\   /ppp1
                     /-----\
                     |     |
                     |  C  |
                     |     |
                     \-----/
                        |eth0
                        |
                   |---------|
                   192.168.3.0 /
                      255.255.255.0
```

A,B,C 세개의 라우터가 있다. 각각은 Class C IP 네트워크(netmask 255.255.25 5.0)의 이더넷 세그먼트를 지원한다. 또 각 라우터에는 다른 라우터로의 PPP 링 크가 있다. 네트워크는 삼각형을 이룬다.

라우터 A의 라우팅 테이블은 다음처럼 보일것이 명백하다.

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
   # route add -net 192.168.2.0 netmask 255.255.255.0 ppp0
   # route add -net 192.168.3.0 netmask 255.255.255.0 ppp1
```

이것은 라우터 A 와 B 사이의 링크에 문제가 생기기 전에는 잘 동작할 것이다. 그 링크에 문제가 생기면 위에서 보여준 라우팅 엔트리를 가지고는 A 이더넷 세 그먼트상의 호스트는 이더넷 세그먼트 B 에 있는 호스트에 도달하지 못하는데, 데이터그램이 라우터 A 의 끊어진 ppp0로 보내지기 때문이다. 그들은 여전히 이더넷 세그먼트 C 의 호스트로는 말할수 있으며, C 이더넷 세그먼트의 호스트도 여전히 B 이더넷 세그먼트상의 호스트와 이야기를 할수있는데, B 와 C 사이의 링크가 여전히 끊어지지 않고 있기 때문이다.

잠깐, A 가 C 로 말할수 있고, C 가 B 로 말할수 있다면 어째서 A 는 C 가 B로 보내도록 C를 경우하여 데이터그램을 B로 보내지 않는가? 이런것이 바로 RIP 같 은 다이나믹 라우팅 프로토콜이 해결하도록 설계된 그런 문제이다. A, B, C 라 우터가 각각 라우팅 데몬은 운영할 때, 링크중 하나에 문제가 생긴다면 라우팅 테이블은 새로운 네트워크 상황은 반영하여 자동으로 적용될것이다. 이런 네트 워크를 설정하는 것은 간단하며, 각 라우터에 대해 두가지만 해주면 된다. 이 경우 라우터 A 에 대하여,

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
   # /usr/sbin/routed
```

’routed’ 라우팅 데몬은 시작할때 자동으로 활성화된 모든 네트워크 포트를 찾 으며 그 호스트의 라우팅 테이블을 결정하고 업데이트 할 수 있도록 각 네트워 크 디바이스에 메세지를 보내고 또 듣는다.

이것은 여러분이 다이나믹 라우팅 프로토콜을 사용할 수 있는 곳에 대한 간단한 설명이다. 더 자세한 정보를 원한다면 문서의 첫부분에 제시된 참고문헌을 보기 바란다.

다이나믹 라우팅에 관하여 중요한 점은..

1. 여러분의 리눅스 머신이 목적지를 향한 다양한 루트가 있을때에만 다이나믹 라우팅 프로토콜을 돌릴 필요가 있다.
2. 다이나믹 라우팅 데몬은 네트워크의 변화에 적응하여 자동으로 라우팅 테이 블을 변경할 것이다.
3. RIP는 작은 또는 중간 사이즈의 네트워크에 적합하다.

5.8 네트워크 서버와 서비스의 설정.

네트워크 서버와 서비스는 원격 사용자가 여러분의 리눅스 머신을 사용할수있게 하기위한 프로그램이다. 원격 사용자는 여러분의 머신, 서버 프로그램 또는 네 트워크 데몬으로 커넥션을 성립하여 해당 포트가 커넥셔을 받아들인후 실행을 한다. 네트워크 데몬이 동작하는 방법에는 두가지가 있다.

standalone

네트워크 데몬은 지정된 네트워크 포트의 메세지를 들으며 들어오는 커 넥션이 성립되었을때 네트워크 커넥션을 관리하고 서비스를 제공한다.

slave to the inetd server

> inetd 서버는 들어오는 커넥션을 전문으로 취급하는 특별한 네트워크 데 몬 프로그램이다. tcp 혹은 udp와 서비스 포트의 특별한 조합이 들어왔 을때, 어떤 프로그램이 실행되야 하는가를 말해주는 설정 파일을 가지고 있다. 이 포트는 다른 파일에 기술되어 있으며 이에 관해서도 곧 이야기 를 할것이다.

여기 설정해야될 두개의 중요한 파일이 있다. 네임을 포트넘버로 할당하는 /etc/services 파일과 inetd 네트워크 데몬의 설정을 위한 /etc/inetd.conf 파 일이다.

5.8.1 /etc/services

/etc/services 파일은 인간에게 친숙한 서비스 네임과 기계에 친숙한 포트넘버 와 관계있는 간단한 데이터베이스 파일이다. 이것의 포맷은 매우 간단하다. 이 것은 각 라인이 데이터베이스의 엔트리를 나타내는 텍스트 파일이다. 각각의 엔 트리는 몇개의 공백 스페이스(tab 또는 space)로 나뉘는 세개의 필드로 구성되 어 있다. 이 필드는 아래와 같다.

```
name        port/protocol        aliases          #comment
```

name

기술되는 서비스를 나타내는 간단한 단어이다.

port/protocol

이 필드는 두개의 하위 필드로 나뉜다.

port

지정된 서비스가 가능한 포트넘버를 명시하는 숫자이다. 공통으로 사용하는 서비스에는 할당된 서비스 넘버가 있다. 이들은 RFC-1340 에 나와있다.

protocol

이 서브필드는 tcp나 udp로 세팅된다.

18/tcp 와 18/udp 라는 엔트리는 서로 다른것임을 구분하며, 왜 같은 서 비스가 두곳에 모두 존재해야 하는지에 대한 기술적인 이유가 없음을 알 아두는것이 중요하다. 보통 일반적인 생각이 널리 퍼져있고, 특정 서비 스가 tcp, udp 로 모두 가능하다면 그저 둘 모두에 대한 엔트리를 볼수 있을 뿐이다.

aliases

이 서비스 엔트리를 나타내기 위해 사용되는 다른 이름

'#' 뒤에 나타나는 텍스트는 무시되며 주석으로 간주된다.

/etc/services 파일의 예제. 요즘의 리눅스 배포본은 훌륭한 /etc/services 파일을 제공한다. 오직 완전히 새로운 머신을 만들어야 할때, 데비안 배포본에서 제공하는 /etc/services 파일 이 있다.

```
# /etc/services:
  # $Id: services,v 1.3 1996/05/06 21:42:37 tobias Exp $
  #
  # Network services, Internet style
  #
  # Note that it is presently the policy of IANA to assign a single well-kn
own
  # port number for both TCP and UDP; hence, most entries here have two ent
ries
```

```
  # even if the protocol doesn't support UDP operations.
  # Updated from RFC 1340, ``Assigned Numbers'' (July 1992).  Not all ports
  # are included, only the more common ones.
tcpmux          1/tcp                           # TCP port service multip
lexer
  echo            7/tcp
  echo            7/udp
  discard         9/tcp           sink null
  discard         9/udp           sink null
  systat          11/tcp          users
  daytime         13/tcp
  daytime         13/udp
  netstat         15/tcp
  qotd            17/tcp          quote
  msp             18/tcp                          # message send protocol
  msp             18/udp                          # message send protocol
  chargen         19/tcp          ttytst source
  chargen         19/udp          ttytst source
  ftp-data        20/tcp
  ftp             21/tcp
  ssh             22/tcp                          # SSH Remote Login Protoc
ol
  ssh             22/udp                          # SSH Remote Login Protoc
ol
  telnet          23/tcp
  # 24 - private
  smtp            25/tcp          mail
  # 26 - unassigned
  time            37/tcp          timserver
  time            37/udp          timserver
  rlp             39/udp          resource        # resource location
  nameserver      42/tcp          name            # IEN 116
  whois           43/tcp          nicname
  re-mail-ck      50/tcp                          # Remote Mail Checking Pr
otocol
  re-mail-ck      50/udp                          # Remote Mail Checking Pr
otocol
  domain          53/tcp          nameserver      # name-domain server
  domain          53/udp          nameserver
  mtp             57/tcp                          # deprecated
  bootps          67/tcp                          # BOOTP server
  bootps          67/udp
  bootpc          68/tcp                          # BOOTP client
  bootpc          68/udp
  tftp            69/udp
  gopher          70/tcp                          # Internet Gopher
  gopher          70/udp
  rje             77/tcp          netrjs
  finger          79/tcp
  www             80/tcp          http            # WorldWideWeb HTTP
```

```
  www           80/udp                          # HyperText Transfer Prot
ocol
  link          87/tcp          ttylink
  kerberos      88/tcp          kerberos5 krb5  # Kerberos v5
  kerberos      88/udp          kerberos5 krb5  # Kerberos v5
  supdup        95/tcp
  # 100 - reserved
  hostnames     101/tcp         hostname        # usually from sri-nic
  iso-tsap      102/tcp         tsap            # part of ISODE.
  csnet-ns      105/tcp         cso-ns          # also used by CSO name s
erver
  csnet-ns      105/udp         cso-ns
  rtelnet       107/tcp                         # Remote Telnet
  rtelnet       107/udp
  pop-2         109/tcp         postoffice      # POP version 2
  pop-2         109/udp
  pop-3         110/tcp                         # POP version 3
  pop-3         110/udp
  sunrpc        111/tcp         portmapper      # RPC 4.0 portmapper TCP
  sunrpc        111/udp         portmapper      # RPC 4.0 portmapper UDP
  auth          113/tcp         authentication tap ident
  sftp          115/tcp
  uucp-path     117/tcp
  nntp          119/tcp         readnews untp   # USENET News Transfer Pr
otocol
  ntp           123/tcp
  ntp           123/udp                         # Network Time Protocol
  netbios-ns    137/tcp                         # NETBIOS Name Service
  netbios-ns    137/udp
  netbios-dgm   138/tcp                         # NETBIOS Datagram Servic
e
  netbios-dgm   138/udp
  netbios-ssn   139/tcp                         # NETBIOS session service
  netbios-ssn   139/udp
  imap2         143/tcp                         # Interim Mail Access Pro
to v2
  imap2         143/udp
  snmp          161/udp                         # Simple Net Mgmt Proto
  snmp-trap     162/udp         snmptrap        # Traps for SNMP
  cmip-man      163/tcp                         # ISO mgmt over IP (CMOT)
  cmip-man      163/udp
  cmip-agent    164/tcp
  cmip-agent    164/udp
  xdmcp         177/tcp                         # X Display Mgr. Control
Proto
  xdmcp         177/udp
  nextstep      178/tcp         NeXTStep NextStep       # NeXTStep window
  nextstep      178/udp         NeXTStep NextStep       # server
  bgp           179/tcp                         # Border Gateway Proto.
  bgp           179/udp
```

```
  prospero        191/tcp                         # Cliff Neuman's Prospero
  prospero        191/udp
  irc             194/tcp                         # Internet Relay Chat
  irc             194/udp
  smux            199/tcp                         # SNMP Unix Multiplexer
  smux            199/udp
  at-rtmp         201/tcp                         # AppleTalk routing
  at-rtmp         201/udp
  at-nbp          202/tcp                         # AppleTalk name binding
  at-nbp          202/udp
  at-echo         204/tcp                         # AppleTalk echo
  at-echo         204/udp
  at-zis          206/tcp                         # AppleTalk zone informat
ion
  at-zis          206/udp
  z3950           210/tcp         wais            # NISO Z39.50 database
  z3950           210/udp         wais
  ipx             213/tcp                         # IPX
  ipx             213/udp
  imap3           220/tcp                         # Interactive Mail Access
  imap3           220/udp                         # Protocol v3
  ulistserv       372/tcp                         # UNIX Listserv
  ulistserv       372/udp
  #
  # UNIX specific services
  #
  exec            512/tcp
  biff            512/udp         comsat
  login           513/tcp
  who             513/udp         whod
  shell           514/tcp         cmd             # no passwords used
  syslog          514/udp
  printer         515/tcp         spooler         # line printer spooler
  talk            517/udp
  ntalk           518/udp
  route           520/udp         router routed   # RIP
  timed           525/udp         timeserver
  tempo           526/tcp         newdate
  courier         530/tcp         rpc
  conference      531/tcp         chat
  netnews         532/tcp         readnews
  netwall         533/udp                         # -for emergency broadcas
ts
  uucp            540/tcp         uucpd           # uucp daemon
  remotefs        556/tcp         rfs_server rfs  # Brunhoff remote filesys
tem
  klogin          543/tcp                         # Kerberized `rlogin' (v
5)
  kshell          544/tcp         krcmd           # Kerberized `rsh' (v5)
  kerberos-adm    749/tcp                         # Kerberos `kadmin' (v5)
```

```
  #
  webster         765/tcp                         # Network dictionary
  webster         765/udp
  #
  # From ``Assigned Numbers'':
  #
  #> The Registered Ports are not controlled by the IANA and on most system
s
  #> can be used by ordinary user processes or programs executed by ordinar
y
  #> users.
  #
  #> Ports are used in the TCP [45,106] to name the ends of logical
  #> connections which carry long term conversations.  For the purpose of
  #> providing services to unknown callers, a service contact port is
  #> defined.  This list specifies the port used by the server process as i
ts
  #> contact port.  While the IANA can not control uses of these ports it
  #> does register or list uses of these ports as a convienence to the
  #> community.
  #
  ingreslock      1524/tcp
  ingreslock      1524/udp
  prospero-np     1525/tcp                # Prospero non-privileged
  prospero-np     1525/udp
  rfe             5002/tcp                # Radio Free Ethernet
  rfe             5002/udp                # Actually uses UDP only
  bbs             7000/tcp                # BBS service
  #
  #
  # Kerberos (Project Athena/MIT) services
  # Note that these are for Kerberos v4, and are unofficial.  Sites running
  # v4 should uncomment these and comment out the v5 entries above.
  #
  kerberos4       750/udp         kdc     # Kerberos (server) udp
  kerberos4       750/tcp         kdc     # Kerberos (server) tcp
  kerberos_master 751/udp                 # Kerberos authentication
  kerberos_master 751/tcp                 # Kerberos authentication
  passwd_server   752/udp                 # Kerberos passwd server
  krb_prop        754/tcp                 # Kerberos slave propagation
  krbupdate       760/tcp         kreg    # Kerberos registration
  kpasswd         761/tcp         kpwd    # Kerberos "passwd"
  kpop            1109/tcp                # Pop with Kerberos
  knetd           2053/tcp                # Kerberos de-multiplexor
  zephyr-srv      2102/udp                # Zephyr server
  zephyr-clt      2103/udp                # Zephyr serv-hm connection
  zephyr-hm       2104/udp                # Zephyr hostmanager
  eklogin         2105/tcp                # Kerberos encrypted rlogin
  #
  # Unofficial but necessary (for NetBSD) services
```

```
  #
  supfilesrv      871/tcp                 # SUP server
  supfiledbg      1127/tcp                # SUP debugging
  #
  # Datagram Delivery Protocol services
  #
  rtmp            1/ddp                   # Routing Table Maintenance Proto
col
  nbp             2/ddp                   # Name Binding Protocol
  echo            4/ddp                   # AppleTalk Echo Protocol
  zip             6/ddp                   # Zone Information Protocol
  #
  # Debian GNU/Linux services
  rmtcfg          1236/tcp                # Gracilis Packeten remote config
server
  xtel            1313/tcp                # french minitel
  cfinger         2003/tcp                # GNU Finger
  postgres        4321/tcp                # POSTGRES
  mandelspawn     9359/udp        mandelbrot      # network mandelbrot
# Local services
```

5.8.2 /etc/inetd.conf

/etc/inetd.conf 파일은 inetd 서버의 설정을 위한 파일이다. 이것의 기능은 특 정 서비스에 대한 접속요청이 들어왔을때 무엇을 해야하는지를 inetd에게 알려 주는 것이다. 받아들이기를 원하는 각 서비스에 대해 inetd에게 어떤 서버데몬 은 어떻게 실행할것 인지를 알려주어야 한다.

이것의 포맷은 매우 간단하다. 각 라인이 제공하고픈 서비스를 나타내주는 텍스 트 파일이다. '#' 뒤의 텍스트는 무시되며 주석으로 간주된다. 각 라인은 공백 으로(tab 또는 space)로 구분되는 7개의 필드를 갖는다. 일반적인 포맷은 다음 과 같다.

```
 service  socket_type   proto  flags   user  server_path  server_args
```

service

/etc/services 파일에서처럼 설정과 관계된 서비스이다.

socket_type

이 필드는 해당 엔트리다 관계된다고 간주할 소켓의 타입을 기술한다. 가능한 값은 stream, dgram, raw, rdw 또는 seqpacket이다. 이것은 자연 히 약간 기술적인데, 첫째가는 규칙으로 거의 모든 tcp 기반의 서비스는 stream을 사용하고 거의 모든 udp 기반의 서비스는 dgram을 사용한다. 다른 값을 사용하는것은 매우 특별한 형태의 서버일때 뿐이다.

proto

이 엔트리에 유효하다고 간주되는 프로토콜. 이것은 /etc/services 파일 의 적절한 엔트리와 매치되야 하며 전형적으로 tcp 또는 udp 중의 하나 이다.Sun RPC(Remote Procedure Call)기반의 서버는 rpc/tcp 또는 rpc/udp를 사용한다.

flags

이 필드를 위한 세팅에는 두가지 값밖에 없다. 프로그램을 실행한뒤 소 켓을 놓아주어 다음의 커넥션 요청에 대해 새로운것을 시작하게 하는냐, 아니면 기다리며 다른 서버가 이미 동작한다고 가정하여

다음 터넥션을 다루게 할것이냐하는 두가지이다. 또 이것을 다루는 것은 약간 애매하지 만, tcp 서버는 이 엔트리를 nowait으로 설정하고 udp 서버는 이것을 wait으로 설정하는것이 첫째가는 룰이다. 이것에 예외가 있음을 주의하 고 그러므로 확실하지 않을경우 예제 가이드를 참고하라.

user

네트워크 데몬이 시작할때 /etc/passwd의 어느 계정이 이 데몬의 소유자 가 될것인가를 기술해준다. 이것은 보안문제에 대해 보안장치를 원할때 유용하다. 이 엔트리의 유저를 nobody로 함으로써 네트워크 서버의 보안 이 깨졌을때 피해를 최소화 할수있다. 그러나 이 필드는 전형적으로 root로 설정되는데 많은 서버가 적절히 동작하기 위해서 root의 권한을 요구하기 때문이다.

server_path

이 엔트리에 대해 실행할 실제 서버 프로그램의 경로이다.

server_args

이 필드는 라인의 나머지 부분을 이루며 선택적이다. 이 필드는 서버 데몬 프로그램이 실행될때 프로그램에 넘겨주고싶은 커맨드 라인 변수를 넣어주는 부분이다.

/etc/inetd.conf의 예제. /etc/services 파일에 관해서 처럼 모든 현대의 배포본은 훌륭한 /etc/inetd.co nf 파일을 포함하고 있다. 여기에 데비안 배포본에 포함되어 있는 완전한 /etc/ inetd.conf 파일의 예제가 있다.

```
# /etc/inetd.conf:  see inetd(8) for further informations.
  #
  # Internet server configuration database
  #
  #
  # Modified for Debian by Peter Tobias <<url url="mailto:tobias@et-inf.fho-emden.de" name="tobias@et-i
  #
  # <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
  #
  # Internal services
  #
  #echo           stream  tcp     nowait  root    internal
  #echo           dgram   udp     wait    root    internal
  discard         stream  tcp     nowait  root    internal
  discard         dgram   udp     wait    root    internal
  daytime         stream  tcp     nowait  root    internal
  daytime         dgram   udp     wait    root    internal
  #chargen        stream  tcp     nowait  root    internal
  #chargen        dgram   udp     wait    root    internal
  time            stream  tcp     nowait  root    internal
  time            dgram   udp     wait    root    internal
  #
  # These are standard services.
  #
  telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.teln
etd
  ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.ftpd
  #fsp    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.fspd
  #
  # Shell, login, exec and talk are BSD protocols.
  #
```

```
  shell   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd
  login   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlog
ind
  #exec   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rexe
cd
  talk    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.talk
d
  ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.ntal
kd
  #
  # Mail, news and uucp services.
  #
  smtp    stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.smtp
d
  #nntp   stream  tcp     nowait  news    /usr/sbin/tcpd  /usr/sbin/in.nntp
d
  #uucp   stream  tcp     nowait  uucp    /usr/sbin/tcpd  /usr/lib/uucp/uuc
ico
  #comsat dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.coms
at
  #
  # Pop et al
  #
  #pop-2  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop2
d
  #pop-3  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop3
d
  #
  # ‘cfinger’ is for the GNU finger server available for Debian.  (NOTE: Th
e
  # current implementation of the ‘finger’ daemon allows it to be run as ‘r
oot’.)
  #
  #cfinger stream tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.cfin
gerd
  #finger stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.fing
erd
  #netstat        stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/nets
tat
  #systat stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/ps -auwwx
  #
  # Tftp service is provided primarily for booting.  Most sites
  # run this only on machines acting as "boot servers."
  #
  #tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftp
d
  #tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftp
d /boot
  #bootps dgram   udp     wait    root    /usr/sbin/bootpd        bootpd -i
-t 120
```

```
  #
  # Kerberos authenticated services (these probably need to be corrected)
  #
  #klogin         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin
/in.rlogind -k
  #eklogin        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin
/in.rlogind -k -x
  #kshell         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin
/in.rshd -k
  #
  # Services run ONLY on the Kerberos server (these probably need to be cor
rected)
  #
  #krbupdate      stream tcp      nowait  root    /usr/sbin/tcpd  /usr/sbin
/registerd
  #kpasswd        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin
/kpasswdd
  #
  # RPC based services
  #
  #mountd/1       dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin
/rpc.mountd
  #rstatd/1-3     dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin
/rpc.rstatd
  #rusersd/2-3    dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin
/rpc.rusersd
  #walld/1        dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin
/rpc.rwalld
  #
  # End of inetd.conf.
  ident           stream  tcp     nowait  nobody  /usr/sbin/identd        i
dentd -i
```

5.9 네트워크에 관계된 다른 설정 파일들.

/etc/protocol 파일은 프로토콜 네임을 프로토콜 아이디로 맵핑해주는 데이터베 이스이다. 이것은 프로그래머가 프로그램 내에서 프로토콜을 네임으로 명시할수 있도록 해주며 tcpdump 같은 프로그램이 출력을 넘버대신 이름으로 하도록 해준 다. 이 파일의 일반적인 신택스는 다음과 같다.

```
protocolname number aliases
```

데비안 배포본과 함께 제공되는 /etc/protocols 파일은 다음과 같다.

```
# /etc/protocols:
  # $Id: protocols,v 1.1 1995/02/24 01:09:41 imurdock Exp $
  #
  # Internet (IP) protocols
  #
  #       from: @(#)protocols     5.1 (Berkeley) 4/17/89
```

```
  #
  # Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).
ip      0       IP              # internet protocol, pseudo protocol numb
er
  icmp    1       ICMP            # internet control message protocol
  igmp    2       IGMP            # Internet Group Management
  ggp     3       GGP             # gateway-gateway protocol
  ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ``IP
'')
  st      5       ST              # ST datagram mode
  tcp     6       TCP             # transmission control protocol
  egp     8       EGP             # exterior gateway protocol
  pup     12      PUP             # PARC universal packet protocol
  udp     17      UDP             # user datagram protocol
  hmp     20      HMP             # host monitoring protocol
  xns-idp 22      XNS-IDP         # Xerox NS IDP
  rdp     27      RDP             # "reliable datagram" protocol
  iso-tp4 29      ISO-TP4         # ISO Transport Protocol class 4
  xtp     36      XTP             # Xpress Tranfer Protocol
  ddp     37      DDP             # Datagram Delivery Protocol
  idpr-cmtp       39      IDPR-CMTP       # IDPR Control Message Transport
  rspf    73      RSPF            # Radio Shortest Path First.
  vmtp    81      VMTP            # Versatile Message Transport
  ospf    89      OSPFIGP         # Open Shortest Path First IGP
  ipip    94      IPIP            # Yet Another IP encapsulation
  encap   98      ENCAP           # Yet Another IP encapsulation
```

5.9.1 /etc/networks

/etc/networks 파일은 /etc/hosts 파일과 유사한 기능을 가지고 있다. 이것은 네트워크 네임에 대한 어드레스를 제공하는 간단한 데이터베이스이다. 라인당 오직 두개의 필드를 갖는다는 점만이 다르며 이 필드는 다음과 같이 코딩된다.

```
# networkname networkaddress
```

예제는 다음과 같다:

```
loopnet     127.0.0.0
   localnet    192.168.0.0
   amprnet     44.0.0.0
```

route와 같은 명령을 사용하면, 목적지가 네트워크일고 그 네트워크가 /etc/net works의 엔트리에 있을때 route 명령은 어드레스 대신에 네임을 디스플레이 해 줄것이다.

5.10 네트워크 보안과 억세스 콘트롤.

악의적인 공격에 대해 자신의 머신과 네트워크를 지키는것은 매우 복잡한 기술 이라는것을 경고하며 이 부분을 시작하겠다. 나는 나 자신을 이 분야의 전문가 라고는 생각지 않는다. 또 내가 기술한 다음의 메카니

즘이 도움이 된다 할지라 도 보안문제에 대해 심각하다면 여러분 스스로가 문제에 대해 연구해 볼것을 추천한다. 인터넷에 관련된 문서가 많이 있다.

가장 중요한 룰은 '사용하지 않은 서버는 운영하지 말라!' 이다. 많은 배포판이 모든 종류의 서비스가 설정되고 자동으로 시작하도록 되어있다. 최소레벨의 안 전에 대해서도 확실하게 하고싶다면 /etc/inetd.conf를 잘 살펴보고 사용하지 않을 서비스의 엔트리는 모두 주석처리해라. 좋은 후보감(?)들이 다음의 서비스 들이다.

shell, login, exec, uucp, ftp 그리고 finger, netstat, systat 따위의 정보 서비스이다.

그곳에는 모든 종류의 보안과 억세스 콘트롤 메카니즘이 있으며 나는 그들 대부 분의 요소에 대해 기술할 것이다.

5.10.1 /etc/ftpusers

/etc/ftpusers 파일은 특정 유저가 ftp를 통해 머신으로 들어오는 것을 막아주 는 간단한 메카니즘이다. /etc/ftpusers 는 들어오는 ftp 커넥션이 있을때 ftp 데몬에 의해 읽혀진다. 이 파일은 로그인이 허용되지 않는 사용자들의 간˜Tg˜PA 리스트이다. 이것은 다음과 같이 보일것이다.

```
# /etc/ftpusers - ftp 로그인이 허용안되는 유저들.
   root
   uucp
   bin
   mail
```

5.10.2 /etc/securetty

/etc/securetty는 root 의 로그인이 허용되는 tty 디바이스를 명시하도록 해준 다. /etc/securetty 프로그램은 login 프로그램(흔히 /bin/login)에 의해 읽혀 진다. 이것의 포맷은 root의 로그인이 허용되는 tty 디바이스의 리스트이다. (이외의 tty에서는 root login 불가)

```
#/etc/securetty - tty's on which root is allowed to login
   tty1
   tty2
   tty3
   tty4
```

5.10.3 tcpd 호스트 억세스 콘트롤 메카니즘.

/etc/inetd.conf에 리스트 된것을 보았을 tcpd 프로그램은 보호하도록 설정된 서비스의 로그인과 억세스 콘트롤 메카니즘을 제공한다. 이것이 inetd 프로그램에 의해 호출되었을때, 이것은 억세스를과 적절히 보호하 고 있는 서버를 허용할지 아닐지를 포함하고 있는 두개의 파일을 읽는다.

이것은 첫번째 매치(match)가 발견될때 까지 룰 파일을 찾을것이다. 매치가 발 견 되지 않으면 억세스가 누구에게나 가능한것으로 간주한다. 이것이 찾는 파일 은 순서대로 /etc/hosts.allow /etc/hosts.deny 이다. 나는 이것을 차례로 언급 하겠다. 이 기술의 완벽한 참고를 위해서 여러분은 맨페이지를 참고해야할 필요 가 있다. (hosts_access(5)는 좋은 출발점이 된다.).

/etc/hosts.allow /etc/hosts.allow 파일은 /usr/sbin/tcpd 프로그램의 설정파일이다. hosts.allo w 파일은 어떤 호스트가 여러분 머신상의 서비스에 허용되는지를 기술하는 룰을 포함하는 파일이다.

파일의 포맷은 매우 간단하다.

/etc/hosts.allow # #<service list>: <host list> [: commnad]

service list

이 룰이 적용되는 서버네임의 결정된 리스트의 기호이다. 서버네임의 예는 다음과 같다: ftpd, telnetd, fingerd

host list

호스트 네임의 결정된 리스트의 기호이다. 여기에 IP 어드레스를 사용할 수도 있다. 게다가 그룹의 호스트를 이용하도록 호스트나 IP 어드레스에 와일드 카드를 이용하여 명시할수도 있다. 예를 보자: gw.vk2ktj.ampr.o rg는 명시된 호스트 네임과 매치되고, .uts.edu.au 는 이 스트링으로 끝나는 어떤 호스트네임과도 매치될것이며, 44. 은 이것으로 시작하는 모든 IP와 매치될것이다. 설정은 간단하게 하기위한 몇가지 특별한 표시 가 있는데, 이들중 몇가지는 다음과 같다: ALL은 모든 호스트와 매치되 며, LOCAL은 '.'을 포함하지 않는 모든 호스트와 매치된다. PARANOID는 IP 어드레스와 일치하지 않는 모든 호스트네임과 매치된다(name spoofi ng). 마지막으로 유용한 토큰이 하나 더 있다. EXCEPT는 예외를 갖는 리스트를 제공하는 것을 허용할것이다.

command

이것은 옵션 파라메터이다. 이것은 이 룰이 적용될때마다 실행되는 명령 의 풀 패스네임이다. 이것은 예로써 호스트에 누가 로그인했는가를 구 별하는 명령을 시도할수도 있고, 또는 누군가 접속을 시도한다고 시스템 관리자에게 메일을 보내거나 경고메세지를 보낼수도 있다. 몇가지 포함 될수있는 확장명령이 있는데, 몇가지 흔한 예를 보면: %h는 커넥팅 호스 트의 네임 또는 네임이 아니라면 IP 어드레스를 확징하며, %d는 호출되 는 데몬의 네임을 확장한다.

예:

```
#/etc/hosts.allow
 #
 # Allow mail to anyone
 in.smtpd: ALL
 # All telnet and ftp to only hosts within my domain and my host at home.
 telnetd, ftpd: LOCAL, myhost.athome.org.au
 # Allow finger anyone but keep a record of who they are.
 fingerd: ALL: (finger @%h | mail -s "finger from %h" root)
```

/etc/hosts.deny /etc/hosts.deny 파일은 /usr/sbin/tcpd 프로그램의 설정파일이다. hosts.deny 파일은 여러분 머신상의 서비스에 억세스가 허용되지 않는 호스트를 기록하는 룰을 포함한다.

다음과 같은 간단한 샘플을 볼수있다.

```
# /etc/hosts.deny
   # Disallow all hosts with suspect hostnames
   ALL: PARANOID
   #
   # Disallow all hosts.
   ALL: ALL
```

이경우 다음의 엔트리가 모든 호스트를 막으므로 PARANOID 엔트리는 무의미한 것이다. 이중 하나의 엔트리가 여러분의 요구에 따라 합리적인 디폴트가 될것이 다.

/etc/hosts.deny에 ALL: ALL 디폴트를 갖고 /etc/hosts.allow에 원하는 서비스 와 호스트를 명시하는것이 가장 안전한 설정이다.

5.10.4 /etc/hosts.equiv

hosts.equiv 파일은 특정 호스트와 특정 유저에게 패스워드없이 여러분 머신상 의 계정에 억세스 권한을 주기위해 사용된다. 이것은 여러분이 모든 머신을 제 어할수 있는 안전한 환경에서는 매우 편리하나, 그렇지 않은 경우 보안상의 헛 점이 된다. 여러분의 호스트는 가장 적게 신임받는 호스트만큼만 안전할뿐이다. 보안을 최대화하기 위해서는 이 파일을 사용하지 말고 사용자들도 .rhosts 파일 을 사용하지 않도록 주의를 주어라.

5.10.5 ftp 데몬의 적절한 설정.

많은 사이트들이 사람들로 하여금 특정 아이디를 요구하지 않고 파일을 업 또는 다운로드 할수있도록 익명 ftp를 운영하는데 관심을 두고 있다. 이것을 제공하 기로 결정했다면 anonymous 억세스에 대해 적절한 설정을 해주었는지를 확인하 여라. 대부분의 fptd(8) 맨페이지가 이것에 관해 적당히 기술하고 있다. 여러분 은 항상 이 주의를 따르고 있는지 확인해야 한다. 가장 중요한 팁은 anonymous 계정의 /etc 디렉토리에 /etc/passwd의 복사본을 사용하지 않는것이다. 반드시 필요한 계정을 제외하고는 모두 없애버리도록 해라. 그렇지 않으면 여러분은 패 스워드 크랙킹공격에 대해 많은 약점을 지니게 될것이다.

5.10.6 네트워크 파이어월링.

데이터그램을 여러분의 서버나 머신에 도달하지 못하게 하는것은 뛰어난 보안수 단이다. 이것에 관한 자세한 내용은 firewall-howto에서 다루어진다.

5.10.7 다른 제안들.

여기에 다른, 잠재적으로 여러분이 세심히 고려해봐야 할 제안이 있다.

sendmail

그것의 대중성에도 불구하고 센드메일 데몬은 보안 경고 공고에 대해 상 당히 두려워하는 것처럼 보인다. 그것은 여러분에게 달렸지만, 나라면 센드메일을 운영하지 않겠다.

NFS 와 그외 Sun RPC 서비스.

이것을 경계하라. 이 서비스에는 모든 종류의 exploit이 있다. NFS 같은 서비스 옵션을 발견한다는 것은 상당히 어렵다. 그러나 여러분이 그것을 설정한다면 마운트할 권리를 누구에게 부여하는가에 대해 신중하라.

제 6 절 Network Technology Specific Information.

다음의 하위 부분들은 특정 네트워크 기술에 대해 명시한 것이다. 이 부분에 포 함된 정보는 다른 타입의 네트워크 기술에 반드시 적용되는 것은 아니다.

6.1 ARCnet

ARCNET 디바이스는 'arc0s', 'arc1e', 'arc2e' 등으로 이름붙여진다. 커널에 의 해 처음으로 발견되는 카드는 'eth0'에 할당되며 나머지는 그들이 발견되는 순 서에 따라 차례대로 할당된다. 마지막의 문자는 이더넷 인캡슐레이션 패킷을 선 택했는지, RFC1051 패킷 포맷을 사용했는지를 알려준다.

커널 컴파일 옵션:

Network device support —> [*] Network device support <*> ARCnet support [ ] Enable arc0e (ARCnet "Ether-Encap" packet format) [ ] Enable arc0s (ARCnet RFC1051 packet format)

한번이라도 이더넷 카드의 지원을 위해 커널을 적절히 만들어 본적이 있다면 이 카드의 설정은 쉬울것이다.

전형적으로 다음과 같은것을 사용할 것이다:

```
# ifconfig arc0e 192.168.0.1 netmask 255.255.255.0 up
   # route add 192.168.0.0 netmask 255.255.255.0 arc0e
```

자세한 정보는 /usr/src/linx/Documentation/networking/arcnet-hardware.txt 파일을 참고하라.

ARCNet 지원은 Avery Pennarun, *apenwarr@foxnet.net* `<mailto:apenwarr@foxnet.net>`에 의해서 개발되었다.

6.2 Appletalk(AF_APPLETALK)

Appletalk 지원은 특별한 디바이스를 사용하지 않는데, 이것이 기존의 네트워크 디바이스를 사용하기 때문이다. 이것의 중요한 사용은 리눅스 머신과 애플 컴퓨 터가 프린터나 디스크등의 자원을 공유하기 위해서이다. 추가의 소프트웨어가 필요하며 이것은 netatalk라고 불린다. Wesley Craig *netatalk@umich.edu* `<mailto:netatalk@umich.edu>`가 미 시간 대학의 'Research Systems Unix Groups'이란 팀을 대표했으며 그들은 neta talk 패키지를 개발해 내었는데, 이것은 Appletalk 프로토콜 스택과 유용한 유 틸리티를 제공하는 소프트웨어를 제공한다. netatalk 패키지는 리눅스 배포본에 의해 제공되거나 아니면 미시간 대학에 있는 홈사이트에서 ftp 전송을 받을수있 다.

<*terminator.rs.itd.umich.edu/unix/netatalk/* `<ftp://terminator.rs.itd.umich.edu/unix/netatalk/>`>

이 패키지를 설치하기 위해서는 다음과 같이 하라.

```
# cd /usr/src
   # tar xvfz .../netatalk-1.4b2.tar.Z
   - 여기서 'Makefile'을 수정하고 싶을수도 있는데, 특히 파일들이 어디
```

에 설치될 것인가를 정의해주는 DESTDIR 변수를 바꾸어 주기 위해 그 럴 것이다. 디폴트는 /usr/local/atalk로 되어있는데 매우 안전하다.

```
# make
- as root:
# make install
```

6.2.1 Appletalk 소프트웨어의 설정.

이것이 동작하게 하기위해 가장 먼저 해야할일은 /etc/services 파일에 새로운 엔트리를 추가하는 것이다. 추가할 엔트리는 다음과 같다:

```
rtmp    1/ddp   # Routing Table Maintenance Protocol
   nbp     2/ddp   # Name Binding Protocol
   echo    4/ddp   # AppleTalk Echo Protocol
   zip     6/ddp   # Zone Information Protocol
```

다음은 /usr/local/atalk/etc 디렉토리에(혹은 패키지를 설치한 디렉토리에) ap pletalk 설정파일을 만들어 주는 것이다.

처음으로 만들어줘야 하는 파일은 /usr/local/atalk/etc/atalkd.conf 파일이다. 처음에는 이 파일은 애플 머신이 있는 네트워크를 지원하는 네트워크 디바이스 의 이름을 지정하는 하나의 라인만이 필요할뿐이다.

```
eth0
```

Appletalk 데몬은 나머지 세부사항들을 실행뒤에 추가할것이다.

6.2.2 Appletalk를 경유한 리눅스 파일시스템의 탐험.

네트워크상의 애플머신이 공유하도록 리눅스 파일시스템을 네트워크로 export 할수도 있다.

이것을 위해서는 /usr/local/atalk/etc/AppleVolumes.system 파일을 설정해야 한다. /usr/local/atalk/etc/AppleVolumes.default라 불리는 다른 설정 파일도 있는데 이것은 완전히 같은 포맷을 가지고 있으며, 게스트 권한을 가지고 접속 한 사용자가 어떤 파일 시스템을 받을 것인가를 기술한다.

이 파일의 설정에 대한 모든 세부사항과 어떤 다양한 옵션이 있는가는 afpd 맨 페이지에서 찾을수 있다.

다음과 같은 간단한 예를 볼수있다:

```
/tmp Scratch
   /home/ftp/pub "Public Area"
```

이것은 /tmp 파일시스템을 AppleShare Volume의 'Scratch'로, ftp public 디렉 토리를 AppleShare Volume의 'Public Area'로 export 할것이다. Volume의 이름 은 강제적인 것이 아니고 데몬이 몇가지를 선택하며, 어쨋든 그들을 명시하지 말아라.

6.2.3 Appletalk 와 리눅스 프린터 공유하기.

아주 간단하게 리눅스 프린터를 애플 머신과 공유할수 있다. 여러분은 Apple Pr inter Access Protocol Daemon인 papd를 실행할 필요가 있다. 이 프로그램을 운 영하면 이것은 애플 머신으로 부터 요청을 받아들이고 로컬 라인 프린터 데몬으 로 프린트 잡을 스풀할것이다.

이 데몬의 설정을 위해서는 /usr/local/atalk/etc/papd.conf 파일을 수정할 필 요가 있다. 정의에 넣어준 이름은 Appletalk naming 프로토콜과 함께 등록될것 이다.

다음과 같은 간단한 예를 볼수있다:

```
TricWriter:\
       :pr=lp:op=cg:
```

이것은 Appletalk 네트워크에서 가능한 'TricWriter'란 이름의 프린터를 만들며 들어오는 job은 lpd를 사용하는 'lp'(/etc/printcap에 정의된 바와 같이) 프린 터로 프린트 될것이다. 'op=cg' 엔트리는 리눅스 유저 'cg'가 프린터의 오퍼레 이터임을 말해준다.

6.2.4 Appletalk 소프트웨어 시작하기.

ok, 이제 여러분은 기본 설정을 테스트할 준비가 되었다. netatalk 패키지와 함 께 제공되며 잘 동작하는 rc.atalk 파일이 있다. 그러므로 여러분은 다음만 해 주면 된다.

```
# /usr/local/atalk/etc/rc.atalk
```

모든 시작하고 잘 돌아갈것이다. 아무런 에러메세지도 나지 않을것이고, 각 스 테이지가 시작할때 이 소프트웨어는 콘솔로 메세지를 보낼것이다.

6.2.5 appletalk 소프트웨어의 테스트.

이 소프트웨어가 적절하게 동작하는가를 테스트하기 위해서는 애플머신으로 달 려가서, Apple 메뉴를 풀다운하고, Chooser를 선택한뒤, AppleShare를 클릭해 라. 그러면 리눅스 박스가 보일것이다.

6.2.6 appletalk 소프트웨어의 주의할점.

1. IP 네트워크를 설정하기 전에 appletalk 지원을 시작해야 할지도 모른다. Appletalk 프로그램을 시작하는데 문제가 있다면, 혹은 그것은 시작한뒤에 IP 네트워크에 문제가 생겼다면, /etc/rc.d/rc.inet1 파일을 실행하기 전에 Appletalk 프로그램을 시작해 보아라.
2. afpd(Apple Filing Protocol Daemon) 지독하게 하드디스크를 잡아먹는다. 마 운트 포인트 아래에 이것은 몇개의 디렉토리를 만든다. 이것은 .AppleDouble 디렉토리를 만들어서 resource fork를 저장할 수 있다. 그러므로 '/'를 익스 포팅(exporting)하기 전에 두번 생각하라. 나중에 정리하는데 많은 시간이 걸릴것이다.
3. afpd 프로그램을 맥으로부터 클리어 텍스트 패스워드를 기대한다. 보안이 문제가 될 수 있으므로 인터넷에 연결된 머신에서 이 데몬을 운영할때는 신중하라. 누군가가 못된짓을 한다면 여러분 자신을 비난해야 한다.
4. netstat, ifconfig등 기존의 분석툴은 Appletalk를 지원하지 않는다. 필요하 다면 /proc/net 디렉토리에서 정보를 얻을수 있다.

6.2.7 그외의 정보.

리눅스에서 Aplletalk를 설정하는 방법에 대한 더 자세한 기술은 thehamptons.c om <*thehamptons.com/anders/netatalk/* `<http://thehamptons.com/anders/netatalk/>`>의 Anders Brownworth Linux Net atalk-HOWTO 페이지를 참고하라.

6.3 ATM

리눅스에 Asynchronous Transfer Mode를 지원하기 위한 프로젝트를 Werner Alme sberger <*werner.almesberger@lrc.di.epfl.ch* `<mailto:werner.almesberger@lrc.di.epfl.ch>`>가 관리중이다. 프로젝트의 상황 에 대한 현재 정보는 다음에서 얻을수 있다.<*www.epfl.ch/linux-atm* `<http://www.epfl.ch/linux-atm>`>

6.4 AX25 (AF_AX25)

AX.25 디바이스는 커널 2.0.*에서 'sl0', 'sl1' 등이고, 커널 2.1.*에서 'ax0', 'ax1' 등이다.

Kernel Compile Options: Networking options —> [*] Amateur Radio AX.25 Level 2

AX25, Netrom, Rose 프로토클은 AX25-HOWTO 에서 다루어 진다. 이 프로토콜은 패킷 라디오 실험의 세계 아마추어 라디오 오퍼레이터에 의해 사용된다.

이 프로토콜의 implementation의 대부분의 작업은 Jonathon Naylor, *jsn@cs.not.ac.uk* `<mailto:jsn@cs.not.ac.uk>`에 의해 이루어졌다.

6.5 DECNet.

DECNet의 지원은 현재도 계속 작업중이다. 이후의 2.1.* 커널에서는 볼수도 있 겠다.

6.6 EQL - multiple line traffic equaliser

EQL 디바이스 이름은 'eql'이다. 표준의 커널소스로는 한 머신당 하나의 EQL 디 바이스만을 갖는다. EQL은 tcp/ip를 운송하는 싱글 로지컬 링크로서의 PPP, sli p, plip등의 다중 point to point 활용의 수단을 제공한다. 여러개의 저속 라인 을 쓰는 것이 종종 한개의 고속 라인을 쓰는것보다 더 싸다.

Kernel Compile Options: Networking options —> [*] Amateur Radio AX.25 Level 2

이 메카니즘을 지원하기 위해서는 라인의 다른 끝에 있는 머신도 EQL을 지원해 야 한다. Linux, Livingstone, Portmaster 그리고 새로운 다이얼인 서버가 호환 능력을 제공한다.

EQL을 설정하기 위해서는 다음에서 구할수 있는 eql 툴이 있어야 한다. sunsite.unc.edu <*sunsite.unc.edu/pub/linux/system/Serial/eql-1.2.tar.gz* `<ftp://sunsite.unc.edu/pub/linux/system/Serial/eql-1.2.tar.gz>`>

설정은 대단히 수월하다. eql 인터페이스를 설정하면서 시작한다. eql 인터페이 스는 다른 네트워크 인터페이스와 비슷하다. 다음과 같이 ifconfig 유틸리티를 이용하여 IP 어드레스와 myu를 설정한다.

```
ifconfig eql 192.168.10.1 mtu 1006
   route add default eql
```

다음으로 사용할 각 라인을 수동으로 초기화해줄 필요가 있다. 이것은 point to point 디바이스의 조합으로 나타난다. 커넥션을 어떻게 초기화할것인가 하는것 은 라인의 종류에 따라 달라지며, 더 자세한 정보는 적절한 부분을 참고하기 바 란다.

마지막으로 EQL 디바이스와 시리얼 링크를 연결할 필요가 있으며, 이것은 'ensl aving'이라고 불리고 보이는 바와 같이 eql_enslave 명령으로 이루어진다.

```
eql_enslave eql sl0 28800
   eql_enslave eql ppp0 14400
```

여러분이 eql_enslave에 제공하는 'estimated speed' 파라메터는 직접적으로는 아무것도 하지 않을 것이다. 이것은 EQL 드라이버에 의해 디바이스가 얼만큼의 데이터그램을 받을것인가를 결정하기 위해 사용된다. 그러므로 여러분은 이 값 을 사용함으로써 균형을 적절히 조절할수 있다.

EQL 디바이스로 부터 라인의 연결을 끊기 위해서는 다음과 같이 eql_emancipate 명령을 사용한다.

```
eql_emancipate eql sl0
```

라우트가 실제의 시리얼 디비이스 대신에 eql 디바이스를 참조하는 것을 제외하 면 다른 point to point 링크로 라우팅을 추가할수도 있다. 전형적으로 다음과 같이 하용하겠지:

```
route add default eql0
```

EQL 드라이버는 Simon Janes *simon@ncm.com* `<mailto:simon@ncm.com>`에 의해 개발되었다.

6.7 이더넷.

이더넷 디바이스의 이름은 'eth0', 'eth1' ,'eth2' 등이다. 커널에 의해 첫번째 로 발견되는 카드는 'eth0'에 할당되며 나머지는 발견되는 발견되는 순서대로 할당된다.

리눅스 상에서 이더넷 카드의 동작법을 알고싶다면 Ethernet-HOWTO를 참고하라.

이더넷 카드를 지원하도록 커널을 만들어 보았다면 이 설정은 쉬울것이다.

대체로 다음과 같이 나타날것이다.

```
# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
   # route add 192.168.0.0 netmask 255.255.255.0 eth0
```

대부분의 이더넷 드라이버는 Donald Becker, *becker@CESDIS.gsfc.nasa.gov* `<mailto:becker@CESDIS.gsfc.nasa.gov>`에 의 해 개발되었다.

6.8 FDDI

FDDI의 디바이스 네임은 'fddi0', 'fddi1', 'fddi2' 등이다. 커널에 의해 처음 으로 발견되는 디바이스는 'fddi0'에 할당되고, 나머지는 발견되는 순서대로 할 당된다.

Lawrence V. Stefani, *stefani@lkg.dec.com* `<mailto:stefani@lkg.dec.com>`는 Digital Equipment Corporation FDDI EISA 그리고 PCI 카드를 개발했다.

Kernel Compile Options: Network device support —> [*] FDDI driver support [*] Digital DEFEA and DEFPA adapter support

FDDI 드라이버를 지원하는 커널을 만들고 설치했다면, FDDI 인터페이스의 설정 은 이더넷 인터페이스의 설정과 거의 동일하다. 적절한 FDDI 인터페이스의 이름 을 ifconfig과 route 명령에 명시해야 한다.

6.9 Frame Relay

DLCI 인캡슐레이션 디바이스를 위한 프레임 릴레이의 디바이스 이름은 'dlci00' , 'dlci01' 등이며, FRAD를 위한 것은 'sdla0', 'sdla1'등이다.

프레임 릴레이는 돌발적인 또는 간헐적인 데이커 커뮤니케이션 트래픽에 적용하 기 위해 설계된 새로운 네트워킹 기술이다. Frame Relay Access Device(FRAD)를 사용해 프레임 릴레이 네트워크에 접속할수 있다. 리눅스 프레임 릴레이는 RFC- 1490에 기술된대로 프레임 릴레이 상에서 IP를 지원한다.

Kernel Compile Options: Network device support —> <*> Frame relay DLCI support (EXPERIMENTAL) (24) Max open DLCI (8) Max DLCI per device <*> SDLA (Sangoma S502/S508) support

Mike McLagan, mike.*mclagan@linux.org* `<mailto:mclagan@linux.org>`가 프레임 릴레이의 지원과 설정툴을 개 발했다.

현재 지원되는 유일한 프레임 릴레이는 Sangoam Technology의 S502A, S502E, S5 08이다.

커널의 재설치후에 FRAD와 DLCI를 설정하기 위해서는 ftp.invlogic.com에서 얻 을수 있는 프레임 릴레이 설정툴이 필요하다. <ftp://ftp.invlogic.com/pub/linux/fr/frad-0.15.tgz>. 컴파일과 설치는 매우 수월하나, top 레벨의 Makefile이 없으므로 수동으로 해주어야 한다.

```
# cd /usr/src
   # tar xvfz .../frad-0.15.tgz
   # cd frad-0.15
   # for i in common dlci frad; do cd $i; make clean; make; cd ..;done
   # mkdir /etc/frad
   # install -m 644 -o root -g bin/*.sfm /etc/frad
   # install -m 700 -o root -g root frad/fradcfg /sbin
   # install -m 700 -o root -g root dlci/dlcicfg /sbin
```

툴을 설치한뒤에 /etc/frad/router.conf 파일을 만들 필요가 있다. 이 템플릿을 이용할수도 있으며, 이것은 예제 파일의 수정 버전이다.

```
# /etc/frad/router.conf
  # This is a template configuration for frame relay.
  # All tags are included. The default values are based on the code
  # supplied with the DOS drivers for the Sangoma S502A card.
  #
  # A '#' anywhere in a line constitutes a comment
  # Blanks are ignored (you can indent with tabs too)
  # Unknown [] entries and unknown keys are ignored
  #
[Devices]
  Count=1                 # number of devices to configure
  Dev_1=sdla0             # the name of a device
  #Dev_2=sdla1            # the name of a device
# Specified here, these are applied to all devices, and can be overriden
for
  # each individual board.
  #
  Access=CPE
  Clock=Internal
  KBaud=64
  Flags=TX
  #
  # MTU=1500              # Maximum transmit IFrame length, default is 4096
  # T391=10               # T391 value    5 - 30, default is 10
  # T392=15               # T392 value    5 - 30, default is 15
  # N391=6                # N391 value    1 - 255, default is 6
  # N392=3                # N392 value    1 - 10, default is 3
  # N393=4                # N393 value    1 - 10, default is 4
# Specified here, these set the defaults for all boards
  # CIRfwd=16             # CIR forward   1 - 64
  # Bc_fwd=16             # Bc forward    1 - 512
  # Be_fwd=0              # Be forward    0 - 511
  # CIRbak=16             # CIR backward  1 - 64
  # Bc_bak=16             # Bc backward   1 - 512
  # Be_bak=0              # Be backward   0 - 511
#
  #
  # Device specific configuration
  #
  #
#
  # The first device is a Sangoma S502E
  #
  [sdla0]
  Type=Sangoma            # Type of the device to configure, currently only
                          # SANGOMA is recognised
  #
  # These keys are specific to the 'Sangoma' type
```

```
  #
  # The type of Sangoma board - S502A, S502E, S508
  Board=S502E
  #
  # The name of the test firmware for the Sangoma board
  # Testware=/usr/src/frad-0.10/bin/sdla_tst.502
  #
  # The name of the FR firmware
  # Firmware=/usr/src/frad-0.10/bin/frm_rel.502
  #
  Port=360                # Port for this particular card
  Mem=C8                  # Address of memory window, A0-EE, depending on c
ard
  IRQ=5                   # IRQ number, do not supply for S502A
  DLCIs=1                 # Number of DLCI's attached to this device
  DLCI_1=16               # DLCI #1's number, 16 - 991
  # DLCI_2=17
  # DLCI_3=18
  # DLCI_4=19
  # DLCI_5=20
  #
  # Specified here, these apply to this device only,
  # and override defaults from above
  #
  # Access=CPE            # CPE or NODE, default is CPE
  # Flags=TXIgnore,RXIgnore,BufferFrames,DropAborted,Stats,MCI,AutoDLCI
  # Clock=Internal        # External or Internal, default is Internal
  # Baud=128              # Specified baud rate of attached CSU/DSU
  # MTU=2048              # Maximum transmit IFrame length, default is 4096
  # T391=10               # T391 value    5 - 30, default is 10
  # T392=15               # T392 value    5 - 30, default is 15
  # N391=6                # N391 value    1 - 255, default is 6
  # N392=3                # N392 value    1 - 10, default is 3
  # N393=4                # N393 value    1 - 10, default is 4
#
  # The second device is some other card
  #
  # [sdla1]
  # Type=FancyCard        # Type of the device to configure.
  # Board=                # Type of Sangoma board
  # Key=Value             # values specific to this type of device
#
  # DLCI Default configuration parameters
  # These may be overridden in the DLCI specific configurations
  #
  CIRfwd=64               # CIR forward   1 - 64
  # Bc_fwd=16             # Bc forward    1 - 512
  # Be_fwd=0              # Be forward    0 - 511
  # CIRbak=16             # CIR backward  1 - 64
  # Bc_bak=16             # Bc backward   1 - 512
```

```
  # Be_bak=0              # Be backward   0 - 511
#
  # DLCI Configuration
  # These are all optional. The naming convention is
  # [DLCI_D<devicenum>_<DLCI_Num>]
  #
[DLCI_D1_16]
  # IP=
  # Net=
  # Mask=
  # Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
  # DLCIFlags=TXIgnore,RXIgnore,BufferFrames
  # CIRfwd=64
  # Bc_fwd=512
  # Be_fwd=0
  # CIRbak=64
  # Bc_bak=512
  # Be_bak=0
[DLCI_D2_16]
  # IP=
  # Net=
  # Mask=
  # Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
  # DLCIFlags=TXIgnore,RXIgnore,BufferFrames
  # CIRfwd=16
  # Bc_fwd=16
  # Be_fwd=0
  # CIRbak=16
  # Bc_bak=16
  # Be_bak=0
```

/etc/frad/router.conf 파일을 만들었으면, 실제 디바이스를 설정하는 일만 남 았다. 이것은 보통의 네트워크 디바이스 설정보다 조금 복잡하며, DLCI 인캡슐 레이션 디바이스 전에 FRAD 디바이스를 가져와야 함을 기억해야 한다.

```
 # Configure the frad hardware and the DLCI parameter
   /sbin/fradcfg /etc/frad/router.conf || exit 1
   /sbin/dlcicfg file /etc/frad/router.conf
   # Bring up the FRAD device
   ifconfig sdla0 up
   #
   # Configure the DLCI encapsulation interface and routing
   ifconfig dlci00 192.168.10.1 pointopoint 192.168.10.2 up
   route add 192.168.10.0 netmask 255.255.255.0 dlci00
   #
   ifconfig dlci01 192.168.11.1 pointopoint 192.168.11.2 up
   route add 192.168.11.0 netmask 255.255.255.0 dlci00
   #
   route add default dev dlci00
   #
```

6.10 IP 어카운팅.

리눅스 커널의 IP 어카운팅 feature는 몇몇 네트워크 사용 데이터를 수집하고 분석할수 있게 해준다. 수집된 데이터는 그것이 마지막으로 리셋된 이후에 몇개 의 패킷과 축적된 몇개의 바이트로 이루어진다. 여러분은 자신의 목적에 맞게 각 figure를 분류하는 다양한 룰을 명시할수 있다.

Kernel Compile Options: Networking options —> [*] IP: accounting

커널을 컴파일하고 설치한뒤에 IP 어카운팅을 설정하기 위해서는 ipfwadm 명령 을 사용할 필요가 있다. 선택할수 있는 IP 어카운팅의 분류방법은 많이 있다. 나는 사용하기 좋은 간단한 예제를 하나 선택했으며, 더 자세한 정보를 원하면 ipfwadm 맨페이지를 읽어보기 바란다.

시나리오: PPP로 인터넷에 연결된 이더넷 네트워크가 있다. 이더넷 상에는 **IJI**가 지 서비스를 제공하는 머신이 있고, telnet, rogin, ftp, www 등에 의해 발생하 는 트래픽을 알고싶어 한다.

아마도 다음과 같이 보이는 명령어 셋을 사용할것이다.

```
#
   # Flush the accounting rules
   ipfwadm -A -f
   #
   # Add rules for local ethernet segment
   ipfwadm -A in -a -P tcp -D 44.136.8.96/29 20
   ipfwadm -A out -a -P tcp -S 44.136.8.96/29 20
   ipfwadm -A in -a -P tcp -D 44.136.8.96/29 23
   ipfwadm -A out -a -P tcp -S 44.136.8.96/29 23
   ipfwadm -A in -a -P tcp -D 44.136.8.96/29 80
   ipfwadm -A out -a -P tcp -S 44.136.8.96/29 80
   ipfwadm -A in -a -P tcp -D 44.136.8.96/29 513
   ipfwadm -A out -a -P tcp -S 44.136.8.96/29 513
   ipfwadm -A in -a -P tcp -D 44.136.8.96/29
   ipfwadm -A out -a -P tcp -D 44.136.8.96/29
   ipfwadm -A in -a -P udp -D 44.136.8.96/29
   ipfwadm -A out -a -P udp  -D 44.136.8.96/29
   ipfwadm -A in -a -P icmp -D 44.136.8.96/29
   ipfwadm -A out -a -P icmp -D 44.136.8.96/29
   #
   # Rules for default
   ipfwadm -A in -a -P tcp -D 0/0 20
   ipfwadm -A out -a -P tcp -S 0/0 20
   ipfwadm -A in -a -P tcp -D 0/0 23
   ipfwadm -A out -a -P tcp -S 0/0 23
   ipfwadm -A in -a -P tcp -D 0/0 80
   ipfwadm -A out -a -P tcp -S 0/0 80
   ipfwadm -A in -a -P tcp -D 0/0 513
   ipfwadm -A out -a -P tcp -S 0/0 513
   ipfwadm -A in -a -P tcp -D 0/0
   ipfwadm -A out -a -P tcp -D 0/0
   ipfwadm -A in -a -P udp -D 0/0
   ipfwadm -A out -a -P udp  -D 0/0
   ipfwadm -A in -a -P icmp -D 0/0
```

```
       ipfwadm -A out -a -P icmp -D 0/0
       #
       # List the rules
       ipfwadm -A -l -n
       #
```

마지막의 명령은 각 어카운팅 룰을 리스팅하고 수집된 토달 정보를 보여준다.

IP 어카운팅을 분석할때 주의해야 할 중요한 점은 매치하는 모든 룰에 대한 토 탈이 증가될것이라는 점인데, 그러므로 다른 figure를 얻기 위해서는 적절한 수 학적 처리를 해야할 필요가 있다. 예를 들어서 얼마나 많은 데이터가 ftp, teln et, rlogin, www의 데이터가 아닌 것인지를 알고 싶다면, 모든 포트에 매치되는 룰에서 개별적인(ftp, telnet, rlogin, www의) 토탈을 빼겠다.

```
# ipfwadm -A -l -n
  IP accounting rules
   pkts bytes dir prot source               destination          ports
      0     0 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 20
      0     0 out tcp  44.136.8.96/29       0.0.0.0/0            20 -> *
      0     0 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 23
      0     0 out tcp  44.136.8.96/29       0.0.0.0/0            23 -> *
     10  1166 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 80
     10   572 out tcp  44.136.8.96/29       0.0.0.0/0            80 -> *
    242  9777 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 513
    220 18198 out tcp  44.136.8.96/29       0.0.0.0/0            513 -> *
    252 10943 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> *
    231 18831 out tcp  0.0.0.0/0            44.136.8.96/29       * -> *
      0     0 in  udp  0.0.0.0/0            44.136.8.96/29       * -> *
      0     0 out udp  0.0.0.0/0            44.136.8.96/29       * -> *
      0     0 in  icmp 0.0.0.0/0            44.136.8.96/29       *
      0     0 out icmp 0.0.0.0/0            44.136.8.96/29       *
      0     0 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 20
      0     0 out tcp  0.0.0.0/0            0.0.0.0/0            20 -> *
      0     0 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 23
      0     0 out tcp  0.0.0.0/0            0.0.0.0/0            23 -> *
     10  1166 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 80
     10   572 out tcp  0.0.0.0/0            0.0.0.0/0            80 -> *
    243  9817 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 513
    221 18259 out tcp  0.0.0.0/0            0.0.0.0/0            513 -> *
    253 10983 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> *
    231 18831 out tcp  0.0.0.0/0            0.0.0.0/0            * -> *
      0     0 in  udp  0.0.0.0/0            0.0.0.0/0            * -> *
      0     0 out udp  0.0.0.0/0            0.0.0.0/0            * -> *
      0     0 in  icmp 0.0.0.0/0            0.0.0.0/0            *
      0     0 out icmp 0.0.0.0/0            0.0.0.0/0            *
  #
```

6.11 IP 알리아싱.

하나의 네트워크에 여러개의 IP 어드레스를 설정할 수 있는 몇가지 응용 프로그 램이 있는데 이것은 꽤 쓸

만하다. 인터넷 서비스 제공자는 고객에게 제공하는 w ww이나 ftp에 이 'customized'를 제공한다.

Kernel Compile Options: Networking options —> .... [*] Network aliasing .... <*> IP: aliasing support

IP 알리아스 기능을 포함하여 커널을 컴파일하고 설치한 뒤의 설정은 매우 간단 하다. 알리아스는 실제 네트워크 디바이스와 연관된 가상 네트워크 디바이스로 추가된다. 간간한 이름짓기 규약은 이 디바이스에 <devname>:<virtual dev num> , 예를들어 eth0:0, ppp0:10의 형태로 적용된다.

예로써, 두개의 서로다른 IP 서브넷을 동시에 지원하는 이더넷 네트워크를 가지 고 있다고 가정하고 두개 모두로의 다이렉트 억세스를 할수있다면 다음과 같이 사용할수 있다.

```
#
   # ifconfig eth0:0 192.168.1.1 netmask 255.255.255.0 up
   # route add -net 192.168.1.0 netmask 255.255.255.0 eth0:0
   #
   # ifconfig eth0:1 192.168.10.1 netmask 255.255.255.0 up
   # route add -net 192.168.10.0 netmask 255.255.255.0
   #
```

알리아스를 지우려면 그것의 이름뒤에 간단히 '-'를 추가하면 된다.

```
# ifconfig eth0:0- 0
```

이 알리아스에 관련된 모든 라우트가 자동으로 지워질것이다.

6.12 IP 파이어월

IP 파이어월과 파이어월 관련 이슈는 firewall-howto에서 깊이 다루어진다. IP 파이어월링은 필터링과 지정된 IP로부터의 데이터그램만 허용하는 등의 방법으 로 허가되지 않은 네트워크 억세스로부터 머신을 보호해준다. 세가지 클래스의 룰이 있는데, incoming filtering, outgoing filtering, forwarding filtering 등이다. Incoming 룰은 네트워크 디바이스로부터 받아들이는 데이터그램에 적용 된다. Outgoing 룰은 네트워크 디바이스에 의해 전송되는 데이터그램에 적용된 다. Filterling 룰은 예를들어 라우팅된 데이터그램과 같이 이 머신을 향하지 않지만 받아들이는 데이터그램에 적용된다.

Kernel Compile Options: Networking options —> [*] Network firewalls .... [*] IP: forwarding/gatewaying .... [*] IP: firewalling [ ] IP: firewall packet logging

IP 파이어월 룰의 설정은 ipfwadm 명령을 통해서 이루어진다. 이미 언급했다시 피, 보안은 내 전문분야가 아니며 내가 여러분이 사용할수 있는 예제를 제공한 다 하더라도, 보안이 자신에거 중요하다면 여러분 스스로가 연구하고 개발해야 한다.

IP 파이어월을 사용하는 가장 흔한 경우는 허가되지 않은 네트워크 외부로부터 의 억세스를 막기위하여 리눅스머신을 라우터나 파이어월 게이트웨이로 이용하 는 경우일 것이다.

다음의 설정은 Arnt Gulbrandsen, <*agulbra@troll.no* `<mailto:agulbra@troll.no>`>의 기고에 근거한 것이다.

아래의 예는 이 다이어그램에 그려진것과 같은 리눅스 파이어월/라우터 머신의 파이어월 설정을 설명하고 있다.

```
-                                    -
 \                                   | 172.16.37.0
  \                                  |   /255.255.255.0
   \                  ---------      |
```

```
           |  172.16.174.30 | Linux |      |
       NET =================|  f/w  |------|    ..37.19
           |    PPP         | router|      |  --------
          /                ---------      |--| Mail |
         /                                |  | /DNS |
        /                                 |  --------
       -                                  -
```

다음의 명령은 보통 rc 파일에 들어있어서 시스템이 부팅될때마다 자동으로 실 행될것이다. 최대한의 보안을 위해서 이것은 네트워크 설정이 끝난뒤에 그러나 네트워크가 완전히 준비되기 전에, (파이어월이 리부팅하는 동안에 누군가가 억 세스 권한을 얻는것을 막기위해서는), 수행되어야 한다.

```
#!/bin/sh

# Flush the 'Forwarding' rules table
 # Change the default policy to 'accept'
 #
 /sbin/ipfwadm -F -f
 /sbin/ipfwadm -F -p accept
 #
 #.. and for 'Incoming'
 #
 /sbin/ipfwadm -I -f
 /sbin/ipfwadm -I -p accept

# First off, seal off the PPP interface
 # I'd love to use '-a deny' instead of '-a reject -y' but then it
 # would be impossible to originate connections on that interface too.
 # The -o causes all rejected datagrams to be logged. This trades
 # disk space against knowledge of an attack of configuration error.
 #
 /sbin/ipfwadm -I -a reject -y -o -P tcp -S 0/0 -D 172.16.174.30

# Throw away certain kinds of obviously forged packet packets right away.
 # Nothing should come from multicast/anycast/broadcast addresses.
 #
 /sbin/ipfwadm -F -a deny -o -S 224.0/3 -D 172.16.37.0/24
 #
 # and nothing coming from the loopback network should ever be
 # seen on a wire
 /sbin/ipfwadm -F -a deny -o -S 127.0/8 -D 172.16.37.0/24

# accept incoming SMTP and DNS connections, but only
 # to the Mail/Name Server
 #
 /sbin/ipfwadm -F -a accept -P tcp -S 0/0 -D 172.16.37.19 25 53
 #
 # DNS uses UDP as well as TCP, so allow that too
 # for question to our name server
 #
```

```
 /sbin/ipfwadm -F -a accept -P udp -S 0/0 -D 172.16.37.19 53
 #
 # but not "answers" coming to dangerous ports like NFS and
 # Larry McVoy’s NFS extension. If you run squid, add its port here.
 #
 /sbin/ipfwadm -F -a deny -o -P udp -S 0/0 53 \
        -D 172.16.37.0/24 2049 2050

# answers to other user port are okay
 #
 /sbin/ipfwadm-F -a accept -P udp -S 0/0 53 \
        -D 172.16.37.0/24 53 1024:65535

# Reject incoming connections to identd
 # We use ’reject’ here so that the connecting host is told
 # straight away not to bother continuing, otherwise we’d experience
 # delays while ident timed out.
 #
 /sbin/ipfwadm -F -a reject -o -P tcp -S 0/0 -D 172.16.37.0/24 113

# Accept some common service connections from the 192.168.64 and
 # 192.168.65 networks, they are friends that we trust.
 #
 /sbin/ipfwadm -F -a accept -P tcp -S 192.168.64.0/23 \
        -D 172.16.37.0/24 20:23

# accept and pass thruough anything originating inside
 #
 /sbin/ipfwadm -F -a accept -P tcp -S 172.16.37.0/24 -D 0/0

# deny most other incoming TCP connections, and log them
 # (append 1:1023 if you have problems with ftp not working)
 #
 /sbin/ipfwadm -F -a deny -o -y - tcp -S 0/0 -D 172.16.37.0/24

# ... for UDP too
 #
 /sbin/ipfwadm -F -a deny -o -P udp -S 0/0 -D 172.16.37.0/24
```

훌륭한 파이어월을 설정하는 것은 조금 복잡하다. 이 예는 여러분에게 합리적인 시작점이 될것이다. ipfwadm의 맨페이지는 이 툴을 사용하는 방법에 있어서 도 움을 제공해 줄것이다. 만약 파이어월을 설정하고자 한다면, 믿을만 하다고 생 각하는 소스로 부터 많은 어드바이스를 얻고 네트워크 외부에서 테스크를 해보 기 바란다.

6.13 IPX (AF_IPX)

IPX 프로토콜은 대개 노벨 네트웨어의 로컬 에리어 네트워크 환경에서 활용된 다. 리눅스는 이 프로토콜의 지원을 포함하고 있으며, IPX의 네트워크 엔드포인 트(endpoint) 또는 라우터로 설정될것이다.

Kernel Compile Options: Networking options —> [*] The IPX protocol [ ] Full internal IPX network

IPX 프로토콜과 NCPFS는 IPX-HOWTO에서 깊이 다루어진다.

6.14 IPv6

여러분이 막 IP 네트워크를 이해하기 시작했다고 생각할때 룰은 변경된다! IPv6 는 인터넷 프로토콜의 버전 6에 대한 간단한 기록이다. IPv6는 인터넷 사회의 할당할 어드레스가 부족하다는 우려를 극복하기 위해서 개발되었다. IPv6 어드 레스는 32byte(128bit)의 길이이다. IPv6는 다른 변경사항을 포함하고 있는데 대개는 간소화에 관한것이고 이것이 IPv4에 비해 IPv6를 다루기 쉽게 만들어 준 다.

리눅스는 2.1.* 커널에 IPv6를 지원하고 있지만 완전한것은 아니다.

신세대의 인터넷 기술을 실험하고 싶다면, 또는 이것이 필요하다면, www.terra. net에서 구할수있는 IPv6-FAQ를 읽어보기 바란다. <*www.terra.net/ipv6/* `<http://www.terra.net/ipv6/>`>.

나머지는 7월 23일 까지 올립니다. 이 문서에 대한 버그 리포트는 *sudoer@nownuri.net* `<mailto:sudoer@nownuri.net>`으로 해주시기 바랍니다.